Proyecto Final (Python + Power BI): 75 Temporadas de Fórmula 1 (1950–2024)

Autor: Pedro · Fecha: 14/08/2025

# 1. Resumen Ejecutivo

Este informe documenta el proyecto final del Máster en Análisis de Datos, centrado en el estudio de las 75 temporadas del Campeonato Mundial de Fórmula 1 (1950–2024). El proyecto se compone de dos entregables principales: (1) un cuaderno de EDA en Python y (2) un dashboard operativo en Power BI. El objetivo es demostrar el dominio de todo el flujo de trabajo analítico: ingestión, limpieza, transformación, análisis descriptivo y estadístico, así como la creación de visualizaciones e indicadores en un producto final consumible por negocio.

# 2. Objetivos y Alcance

• Integrar y depurar datos históricos de F1 en un modelo analítico confiable.

• Ejecutar un Análisis Exploratorio de Datos (EDA) exhaustivo en Python (Pandas/NumPy/matplotlib/seaborn).

• Realizar análisis descriptivo y estadístico para responder preguntas de negocio.

• Diseñar un dashboard operativo en Power BI con pestañas temáticas, indicadores y gráficos.

• Documentar el proceso para su reproducibilidad (README.md) y facilitar el traspaso.

# 3. Datos y Fuentes

El conjunto de datos incluye tablas clásicas del ecosistema de Fórmula 1:

• seasons, races, circuits, results, lap\_times, pit\_stops, qualifying, sprint\_results,

drivers, constructors, status, constructor\_standings, driver\_standings, entre otras.

Periodo de estudio: 1950–2024. La granularidad va desde nivel carrera hasta vuelta, lo que requiere estrategias de rendimiento y un modelado en estrella para BI.

# 4. Arquitectura y Herramientas

• Herramientas principales: Visual Studio Code (Python), Power BI Desktop.

• Librerías Python: pandas, numpy, matplotlib, seaborn, scipy, statsmodels (opcional), pyarrow (opcional).

• Estructura de carpetas sugerida:

├─ data/ (raw, processed)

├─ notebooks/ (EDA)

├─ src/ (transformaciones reutilizables)

├─ reports/ (figuras/exportaciones)

└─ bi/ (archivo .pbix, medidas DAX exportadas)

• Modelo de datos: esquema en estrella con hechos (results, lap\_times, pit\_stops) y dimensiones (drivers, constructors, circuits, calendar, races, status).

# 5. Transformación y Limpieza (Python, Pandas, NumPy)

1) Ingesta y tipificación:

• Carga de CSV.

• Conversión de fechas (race\_date, session\_datetime), enteros y floats (puntos, posiciones, duraciones).

• Normalización de claves (driverId, constructorId, raceId) y verificación de unicidad.

2) Limpieza y calidad:

• Eliminación de duplicados e imputación conservadora de nulos.

• Homogeneización de nombres (drivers/constructors) y estandarización de países.

• Control de outliers (p.ej., duraciones de pit stop anómalas) con reglas estadísticas (IQR) o winsorizing.

3) Enriquecimiento:

• Derivados: temporada (year), era (p.ej., V6 Turbo Híbrida ≥ 2014), región del circuito, métricas de carrera (DNF, puntos normalizados por formato, posiciones ganadas).

• Tablas de calendario (date, year, month, quarter, week, is\_weekend) para orden y agregación.

4) Validación:

• Pruebas de consistencia entre tablas (races ↔ results, results ↔ drivers/constructors).

• Chequeos de cobertura temporal y conteos cruzados (carreras por año, vueltas registradas, etc.).

# 6. Análisis Exploratorio de Datos (EDA)

• Distribuciones: puntos por piloto/constructor, duraciones de pit stop, posiciones de salida y llegada.

• Tendencias: evolución del número de carreras por temporada, puntos medios por GP, variabilidad por era.

• Relaciones: correlación entre posición de clasificación y resultado final; relación entre paradas y posición.

• Calidad de datos: tasa de nulos por campo, duplicados detectados y resoluciones aplicadas.

• Visualizaciones EDA (Python): histogramas, boxplots, violines, barras apiladas por década, dispersión start\_position vs finish\_position con ajuste lineal.

# 7. Buenas Practicas

- Modelo en estrella, desactivar Auto Date/Time, relaciones 1:\* claras.

- Columnas de orden (MesN) para ordenar 'Mes' por número.

- Medidas DAX para KPIs (evitar columnas calculadas innecesarias).

- Segmentaciones sincronizadas (Año, nameGP, Era) y bookmarks para historias.

- Rendimiento: reducir cardinalidad (keys enteras), usar agregaciones en tablas de hechos granulares (lap\_times).

# 9. Medidas DAX Clave (Extracto)

Ejemplos de medidas DAX:

Total Carreras = DISTINCTCOUNT( races[raceId] )

Pilotos Únicos = DISTINCTCOUNT( drivers[driverId] )

Constructores Únicos = DISTINCTCOUNT( constructors[constructorId] )

Puntos Totales = SUM( results[points] )

Podios = CALCULATE( COUNTROWS(results), results[position] <= 3 )

DNF % = DIVIDE( CALCULATE(COUNTROWS(results), status[status] <> "Finished"), COUNTROWS(results) )

Posiciones Ganadas = SUMX( results, results[grid] - results[position] )

Media Pit Stop (s) = AVERAGE( pit\_stops[duration\_s] )

Año Seleccionado = SELECTEDVALUE( Calendar[Year] )

Mes = FORMAT( Calendar[Date], "mmmm" )

Mes Nº = MONTH( Calendar[Date] ) // Usar este campo para ordenar 'Mes'

# 10. Resultados y Hallazgos

• Resumen (ilustrativo):

- Evolución del calendario: incremento del número de carreras por temporada en las últimas décadas.

- Influencia de la clasificación: correlación negativa moderada entre posición de salida y final.

- Paradas en boxes: relación no lineal con el resultado final, dependiente de la era y del circuito.

- Dominio por eras: concentración de victorias en determinados constructores según reglamentos.

Nota: Para evitar sesgos, todas las conclusiones se acompañan de intervalos de confianza y pruebas de significación cuando procede, y se interpretan en el contexto de cambios reglamentarios y de formato.

# 11. Conclusiones

El proyecto demuestra el ciclo completo de analítica con Python y Power BI sobre un caso real y exigente. Se aplicaron principios de calidad de datos, técnicas estadísticas y visualización efectiva para convertir datos históricos de F1 en información accionable. Como trabajo futuro: incorporar telemetría, enriquecer con datos meteorológicos, y explorar modelos predictivos (p.ej., probabilidad de podio) en tiempo real.

# 12. Lecciones Aprendidas

• Modelado en estrella simplifica DAX y mejora rendimiento.

• Separar lógica de limpieza (Python) de lógica semántica (DAX) acelera iteraciones.

• Validaciones automatizadas (tests) evitan regresiones en calidad de datos.

• Mantener convenciones de nombres (snake\_case en Python, PascalCase en DAX) ayuda a la mantenibilidad.

# 13. Anexos

A. Diccionario de tablas (extracto):

• drivers(driverId, code, forename, surname, nationality)

• constructors(constructorId, name, nationality)

• circuits(circuitId, name, country, length\_km)

• races(raceId, year, round, circuitId, date, name)

• results(resultId, raceId, driverId, constructorId, grid, position, points, statusId)

• pit\_stops(raceId, driverId, stop, lap, duration\_s)

• lap\_times(raceId, driverId, lap, position, time\_s)

• status(statusId, status)

• Calendar(Date, Year, Month, MonthN, Quarter)

B. Archivos entregados:

• /mnt/data/F175Y\_EDA.ipynb (EDA en Python)

• /mnt/data/F175Y\_Dashboard.pbix (Dashboard Power BI)