

Spotify Data Analytics

Proyecto de Ingeniería y Análisis de Datos

Pedro Parada

Data Engineer Junior | Data Analytics

1. Origen del proyecto

Durante años escuché Spotify de forma casi obsesiva: playlists, exploración de géneros, sesiones largas de estudio, etapas marcadas por artistas específicos. Cuando noté que Spotify permite descargar el historial completo de reproducción en formato JSON, descubrí algo fascinante: mi vida musical estaba estructurada en datos.

Ahí nació la idea.

- ¿Qué patrones existen en mi escucha?
- ¿Hay correlaciones entre hora del día y tipo de música?
- ¿Cambio de dispositivo según género?
- ¿Existen ciclos estacionales?
- ¿Se puede modelar mi comportamiento musical como un sistema?

Este proyecto nace en la intersección entre dos mundos que me definen: la música y la estadística.

2. Fuente de datos: ¿Qué devuelve Spotify?

Spotify permite solicitar el historial extendido de reproducción mediante su sistema de privacidad (GDPR export).

Los datos se entregan en múltiples archivos JSON con estructura similar a:

- `endTime` → timestamp de reproducción
- `artistName`
- `trackName`
- `msPlayed`

- `device`
- `country`
- `platform`
- `ip_addr`

Características relevantes:

- Datos semi-estructurados.
- Repetición de valores categóricos.
- Información temporal granular.
- Posibles nulos.
- Información sensible (IP).

Esto planteaba inmediatamente desafíos:

- Normalización.
- Limpieza.
- Estandarización temporal.
- Diseño de modelo dimensional.
- Manejo ético de datos sensibles.

3. Motivación técnica

El objetivo no era solo “graficar cosas”.

Quería diseñar un pequeño Data Warehouse con arquitectura clara, aplicando principios reales de ingeniería de datos:

- Separación de responsabilidades.

- ETL estructurado.
- Modelo dimensional en estrella.
- Uso de surrogate keys.
- Datos almacenados en formato eficiente (Parquet).
- Visualización desacoplada (Power BI).

Es un proyecto personal, pero está diseñado con mentalidad profesional.

4. Arquitectura general

El pipeline sigue una arquitectura ETL clásica:

Extract → Transform → Load → Analyze → Visualize

4.1 Extract

- Lectura de múltiples JSON.
- Validación de estructura.
- Manejo de archivos corruptos.
- Unificación en DataFrame base (pandas).

4.2 Transform

- Conversión de timestamps a datetime.
- Creación de columnas derivadas:
 - Año
 - Mes
 - Día
 - Hora
 - Día de semana

- Limpieza de nulos.
- Estandarización de nombres.
- Eliminación de columnas irrelevantes (ej. audiobooks).
- Tratamiento de IP como dato no explotado (por privacidad).

4.3 Diseño dimensional

Se implementó un **Star Schema**:

Dimensiones:

- `dim_date`
- `dim_device`
- `dim_track`
- `dim_location`

Tabla de hechos:

- `fact_streams`

Cada dimensión posee surrogate keys enteras.

¿Por qué?

Porque en modelos analíticos:

- Las claves naturales son inestables.
- Las claves enteras mejoran joins.
- Permite escalar el modelo.

Esto no era necesario para un dataset pequeño, pero lo implementé por convicción arquitectónica.

5. Almacenamiento: Parquet

El resultado transformado se almacena en formato `.parquet`.

Motivación:

- Columnar.
- Compresión eficiente.
- Ideal para análisis.
- Compatible con herramientas BI.

Aunque podría haber usado CSV, Parquet es estándar en pipelines modernos, además me permitía mantener esta estructura de tablas con ID tipo dataframe de pandas ideal para el modelo relacional que buscaba.

6. Correlaciones y análisis

Una vez estructurado el modelo, el análisis se enfocó en:

6.1 Patrones temporales

- Distribución por hora.
- Distribución por día de semana.
- Estacionalidad mensual.

6.2 Intensidad de consumo

`msPlayed` permitió estimar:

- Reproducciones completas vs parciales.
- Engagement real.
- Artistas con escucha sostenida.

6.3 Correlación dispositivo – horario

Análisis interesante:

- Desktop predominante en horario laboral.
- Mobile dominante en desplazamientos.

6.4 Diversidad musical

Se analizaron:

- Top artistas por duración total.
- Concentración vs dispersión.
- Evolución anual de preferencias.

Más allá del dato trivial, buscaba patrones conductuales.

7. Visualización en Power BI

Power BI fue elegido para:

- Modelado relacional.
- Medidas DAX.
- Dashboard interactivo.
- Drill-down temporal.

El dashboard incluye:

- Overview general.
- Top artistas.
- Tendencias mensuales.
- Tendencial anuales
- Distribución horaria.
- Uso por dispositivo.

- Insights destacados.
-

8. Decisiones de ingeniería

Algunas decisiones intencionales:

- Estructura modular del proyecto.
 - Separación de carga, transformación y validación.
 - Uso de `pathlib`.
 - Manejo explícito de errores.
 - Versionado limpio (sin datos sensibles en GitHub).
 - Screenshots en lugar de subir datasets privados.
-

9. Aprendizajes

- Diseñar modelo dimensional desde cero.
 - Entender limitaciones de datos reales.
 - Resolver conflictos de Git en repositorio público.
 - Migrar lógica tipo ETL tradicional (influenciado por experiencia en migración Pentaho → Python en mi internship en Jp Morgan).
 - Pensar en datos como sistema y no como tabla.
-

10. Proyección futura

Posibles mejoras:

- Automatización con Airflow.

- API de Spotify para enriquecer metadata.
 - Análisis de audio features (danceability, valence, energy).
 - Despliegue como dashboard web.
 - Integración con bases SQL reales.
-

11. Cierre

Mi objetivo profesional es integrarme como Data Engineer Junior o Data Analyst, donde pueda seguir desarrollando pipelines reales, trabajar con modelos dimensionales y contribuir en equipos donde los datos no sean solo registros, sino historias estructuradas.