Práctica 1: ELT con Apache Spark en capas Bronze y Silver

Procesamiento Masivo de Datos y Visualización (PMDV)

Curso 2025-2026

1. Introducción

Una vez familiarizados con la arquitectura y herramientas principales de nuestro entorno Big Data, en esta segunda práctica se presenta el caso de uso y conjuntos de datos asociados con los que se trabajará el resto del curso.

El **objetivo principal** es cargar manualmente estos conjuntos datos en la capa *bronze* nuestro Data Lake (MinIO) y, tras esto, usar Apache Spark para transformarlos (limpieza, consolidación, tipado,…) y cargarlos en la capa *silver*, usando el formato Parquet, optimizado para su posterior consumo o procesamiento en el Data Lake.

2. Descripción de la práctica

Como esta asignatura no pone el foco en los procesos de ingesta de datos, **todos los archivos de datos fuente se proporcionarán a través de Moodle y son los únicos que deben usarse**. También se proporcionarán los enlaces web específicos a cada conjunto de datos, pero **solo para la consulta de su significado y otros metadatos** relevantes para su procesamiento y análisis.

El **caso de uso que se propone** es el procesamiento y análisis de los datos **incidentes policiales de la ciudad de San Francisco**. Como otras muchas ciudades del mundo, San Francisco cuenta con un portal de datos abiertos https://data.sfgov.org/ donde puede obtenerse ese conjunto de datos y muchos otros más.

Los datos de incidentes policiales contienen, entre otros, el identificador del incidente, las fechas de ocurrencia y reporte, el tipo y categoría del incidente, el distrito policial, el barrio analítico o la localización geográfica exacta.

Como parte del caso de uso, se requiere analizar estos incidentes en relación con otras variables socioeconómicas. En particular con el ingreso medio de los hogares por zona, disponible en otra web de datos abiertos de EEUU a nivel estatal. Sin embargo, este conjunto de datos de ingresos no está disponible directamente a nivel de barrio, sino a nivel de Census Tract, por lo que es necesario introducir un tercer dataset que actúa como tabla puente, permitiendo relacionar los Census Tracts con los barrios analíticos definidos por el Ayuntamiento de San Francisco.

De este modo, en esta primera práctica **trabajaréis con tres conjuntos de datos que deberéis limpiar y preparar individualmente** para su posterior utilización en prácticas posteriores. Para ello tenéis que **usar Apache Spark** (con Python y Dataframes), ejecutado **desde Notebooks de Jupyter Lab**. Además de los materiales vistos en las sesiones de teoría, puedes consultar la documentación oficial de Spark.

- https://spark.apache.org/docs/4.0.1/api/python/reference/index.html

2.1 Ingesta manual de los datasets en la capa bronze

El primer paso es **descargar cada uno de los 3 conjuntos de datos exclusivamente desde Moodle**, descomprimirlos en caso necesario y cargarlos cada uno en una carpeta distinta dentro del bucket *bronze* de tu Data Lake (MinIO).

A continuación, se enumera cada conjunto de datos a cargar, junto con la URL de la web donde puede **consultarse su definición y estructura**, no los descarguéis desde allí:

- **Datos de incidentes policiales de San Francisco**
  - sf_police_incidents_2018_to_2025_1208.zip
  - https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-2018-to-Present/wg3w-h783/about_data
- **Datos de ingresos medios de los hogares de San Francisco por zona Censal**
  - acs_us_median_household_income_2019-2023.zip
  - En MinIO carga únicamente ACSST5Y2023.S1901-Data.csv , los otros dos son archivos de información adicional (metadatos).
  - https://data.census.gov/table/ACSST1Y2024.S1901?q=S1901+ACS+5-year+Income&g=040XX00US06_050XX00US06075,06075$1400000
- **Datos sobre los barrios analíticos de San Francisco y relación con las zonas censales**
  - 2020_census_tracts_to_neighborhoods_20251208.csv
  - https://data.sfgov.org/Geographic-Locations-and-Boundaries/Analysis-Neighborhoods/j2bu-swwd/about_data

2.2 Procesamiento de cada Dataset y carga en la capa silver

Una vez cargados en la capa *bronze* los conjuntos de datos en su formato original, se requiere aplicar **transformaciones para su limpieza y almacenamiento optimizado** en la capa *silver*, representado por el bucket del mismo nombre que ya creamos en la práctica 0.

Las **transformaciones se deben aplicar de forma individual** para cada conjunto de datos, para lo que se ha de usar **1 notebook de Jupyter Lab con código PySpark por cada uno de los conjuntos de datos**. Esos 3 notebooks formarán

parte de los entregables de la práctica, enumerados en el apartado *3. Entrega de la Práctica*.

Además, se requiere **contestar a diversas preguntas sobre las operaciones o el análisis** del propio conjunto de datos. Para ello **se usará el mismo notebook**, combinando celdas **de código PySpark con** otras de tipo **Markdown,** para contestar a las preguntas combinando código y explicaciones textuales.

En los siguientes subapartados se enumeran las transformaciones a realizar para cada conjunto de datos.

2.2.1 Datos de incidentes policiales de San Francisco

1. **Leed la carpeta** que contiene el fichero CSV desde la zona *bronze* **sin inferir el esquema**.
    i. Leed a nivel de la carpeta que representa a este dataset, no a nivel individual de archivo csv: Police_Department_Incident_Reports__2018_to_Present_20251208.csv
2. Seleccionad **únicamente** las siguientes columnas:
    i. Incident ID,Incident Number,Row ID,Incident Datetime,Report Datetime,Incident Code,Incident Category,Incident Subcategory,Incident Description,Report Type Code,Resolution,Police District,Analysis Neighborhood,Latitude,Longitude
3. Buscad **una forma alternativa distinta al enfoque anterior** que permita obtener exactamente el mismo resultado, pero **eliminando las columnas que no necesitas.**
    i. **No debe usarse el mismo método que en el paso 2.** Debéis **incluir un párrafo en Markdown con la explicación y el enlace a la documentación oficial** donde se describe el método utilizado.
4. Renombrad todas las columnas del DataFrame resultante utilizando **snake_case:** minúsculas, palabras separadas por guiones bajos (_) y sin espacios ni caracteres especiales (ejemplo, *Incident Datetime* → *incident_datetime*).
5. **Convertid** las columnas con fecha y hora (*incident_datetime* y *report_datetime*) **al tipo *timestamp***. A continuación, mostrad el esquema final para comprobar el tipo de datos.
6. **Convertid** el resto de las **columnas numéricas** (identificadores y coordenadas geográficas) a los tipos de datos más adecuados soportados por Spark en cada caso.
    i. https://spark.apache.org/docs/4.0.1/api/python/reference/pyspark.sql/data_types.html

7. A partir de las columnas de fecha ya convertidas, **cread una nueva columna *reporting_delay_minutes*** que contenga la diferencia en minutos entre *report_datetime* e *incident_datetime*.
8. A partir del valor de *reporting_delay_minutes,* **cread una nueva columna categórica *delay_bucket*** de tipo string, cuyo valor indique explícitamente el rango de minutos al que pertenece cada registro:
   i. *"<0"*
   ii. *"0_10"*
   iii. *"10_60"*
   iv. *"60_1440"*
   v. *">1440"*

   Nota: No deben utilizarse etiquetas semánticas (por ejemplo, "bajo", "medio", "alto"), sino el propio rango como valor de la categoría.

9. **Escritura en la capa *silver***, representada por el bucket del mismo nombre. Persiste el Data Frame final de Spark dentro de la carpeta "*silver/sf_police_incidents/*". En lugar de CSV, **usa el formato Parquet**[1].
10. Por último, **responded** a las siguientes **preguntas analíticas**, aplicando para ello las operaciones que consideres oportunas sobre los dataframes anteriores. **Para cada pregunta**, usa una **1 celda Markdown para copiar la pregunta en el notebook, 1 o más celdas de código Spark para obtener la respuesta y 1 celda Markdown para dar responder textualmente**, explicando o analizando los resultados:
    i. ¿Cuántas filas tienen valores nulos en latitude o longitude?

    ii. ¿Cuántos incident_id aparecen más de una vez en el dataset? Ten en cuenta que un mismo incidente puede aparecer en varias filas.

    iii. ¿Cuántos incidentes tienen un retraso entre 10 y 60 minutos en su procesamiento? Usa exclusivamente la columna *delay_bucket* que creaste previamente.

2.2.2 Datos de ingresos medios por census tracts

1. **Leed la carpeta** que contiene el fichero CSV desde la zona *bronze* **sin inferir el esquema**.
    i. Leed a nivel de la carpeta que representa a este dataset, no a nivel individual de archivo csv: ACSST5Y2023.S1901-Data.csv
2. El fichero incluye filas agregadas (Estado y County) que no son de interés para este análisis. Por ello, es necesario **que filtréis las filas cuyo**

[1] Como veremos más adelante en las clases de teoría, este formato es más óptimo para su uso en el Data Lake, una vez hemos ingestado y procesado los datos sin trasnformar.

GEO_ID comience por "1400000US", es decir los Census Tracts correspondientes a San Francisco.

3. **Cread una nueva columna *census_tract*** de tipo string extrayendo la parte numérica posterior a US de la columna GEO_ID.
    i. Ejemplo: 1400000US06075010101 → 06075010101
4. **Seleccionad** únicamente las siguientes columnas **y renombrad** las que se indican:
    i. census_tract
    ii. S1901_C01_012E → median_household_income_raw
    iii. NAME → tract_name
5. A partir de la columna *median_household_income_raw*, **cread una nueva columna *median_household_income*** de tipo double.
    i. Es importante tener en cuenta que existen valores especiales del tipo 250,000+, los cuales deben convertirse a 250000.
6. También a partir de la columna original *median_household_income_raw*, **cread una nueva columna *income_is_censored*** de tipo boolean que cumpla:
    i. El valor de *income_is_censored* debe ser true cuando el valor original de *median_household_income_raw* sea 250,000+.
    ii. En cualquier otro caso, el valor debe ser false.
7. **Escritura en la capa *silver***. Persiste el Data Frame final de Spark dentro de la carpeta "*silver/sf_median_household_income/*" usando **formato Parquet**.
8. Por último, **responded** a las siguientes **preguntas analíticas siguiendo el mismo esquema que en el apartado 2.2.1**, aplicando para ello las operaciones que consideres sobre los data frames anteriores.
    i. ¿Cuántos census tracts hay en total tras el filtrado?
    ii. ¿Cuántos census tracts no tienen información de renta (median_household_income nulo)?
    iii. ¿Cuántos tracts tienen la renta censurada (income_is_censored = true)?
    iv. ¿Cuál es el mínimo, máximo y media de median_household_income (ignorando nulos)?

2.2.3 Datos de los barrios analíticos de San Francisco

1. **Leed la carpeta** que contiene el fichero CSV desde la zona *bronze* **sin inferir el esquema**.
    i. Leed a nivel de la carpeta que representa a este dataset, no a nivel individual de archivo csv: 2020_census_tracts_to_neighborhoods_20251208.csv
2. **Selecciona y renombra** las siguientes columnas en un solo paso:
    i. neighborhoods_analysis_boundaries → analysis_neighborhood

   ii. geoid → census_tract
3. **Verificad la validez de la clave geográfica** representada por la columna census_tract mediante las siguientes comprobaciones:
   i. es de tipo **string**
   ii. tiene longitud 11
   iii. conserva correctamente los ceros a la izquierda

   Además de las operaciones en código Spark en 1 o más celdas, añade una celda Markdown explicando el resultado de las 3 comprobaciones.
4. **Escritura en la capa *silver***. Persiste el Data Frame final de Spark dentro de la carpeta "*silver/*sf_neighborhoods_census_tracts*/*" usando **formato Parquet**.
5. Por último, **responded** a las siguientes **preguntas analíticas siguiendo el mismo esquema que en el apartado 2.2.1**, aplicando para ello las operaciones que consideres sobre los dataframes anteriores.
   i. ¿Cuántos census tracts distintos hay?
   ii. ¿Cuántos barrios analíticos distintos hay?
   iii. ¿Puede un barrio estar asociado a varios census tracts?
   iv. ¿Puede un census tract pertenecer a varios barrios?
   v. Justifica brevemente tus respuestas indicando qué tipo de relación existe entre barrios analíticos y census tracts.

3. Entrega de la Práctica

La fecha límite de entrega para esta práctica es el **viernes 6 de marzo**, hasta las **23:59.**

Debéis entregar un fichero llamado **practica1.zip** que incluya:

- **Notebooks**: Un notebook por cada dataset trabajado en la práctica, **ejecutado correctamente e incluyendo la salida de todas las celdas** usando estrictamente los siguientes nombres**:**
  - ***practica1_dataset1_sf_police_incidents.ipynb***
  - ***practica1_dataset2_sf_median_household_income.ipynb***
  - ***practica1_dataset3_sf_neighborhoods_census_tracts.ipynb***
- **Archivo de identificación:** Archivo de texto **alumnos.txt** incluyendo:
  - Nombre y apellidos de alumno 1
  - Nombre y apellidos del alumno 2 (en caso de trabajo en pareja autorizado)
  - DNI
  - Turno de prácticas

La entrega se realizará **exclusivamente a través de la página de Moodle de la asignatura.**

4. Evaluación

La evaluación de la práctica se realizará de forma global, teniendo en cuenta la **correcta preparación de los datasets**, la **implementación de las transformaciones solicitadas** y la **comprensión del proceso seguido**, reflejada tanto en el código como en las explicaciones incluidas en el notebook.

La calificación final se distribuirá del siguiente modo:

- **Dataset 1 – Incidentes policiales**: 45 %
- **Dataset 2 – Ingresos medios por Census Tract**: 35 %
- **Dataset 3 – Barrios analíticos y Census Tracts**: 20 %

En cada apartado (dataset) se valorará:

- Conjunto de datos resultado (por número de instancias) correcto: 2 puntos
- Transformaciones ejecutadas correctamente: 2 puntos
- Lectura de datos desde capa bronze: 1 punto
- Persistencia de datos en silver: 2 puntos
- Respuesta correcta a las preguntas analíticas: 3 puntos

Los notebooks deberán entregarse **completamente ejecutados**, incluyendo la salida de todas las celdas. La ausencia de resultados visibles podrá afectar negativamente a la calificación.

Anexo I

Debido a que las capacidades de df.show() son relativamente limitadas para poder analizar los conjuntos de datos que se están procesando, se propone un código alternativo para mayor comodidad en la lectura de datos.

Este código implementa una función *display(dataframe)* que permite mostrar el dataset y navegar por los datos. Tiene la implicación de que el dataframe se transforma a Pandas, por lo que es adecuado para depurar y entender el proceso, pero **recordar quitar la función** más adelante **cuando despleguéis los procesos con Airflow**.

```
#Funcion para facilitar visualización
from IPython.display import display
import pandas as pd
pd.set_option("display.max_columns", None)   # muestra todas las
columnas

def display_df(
    df,
    n=10,
    columns=None,
    sort_by=None,
    ascending=False
):
    """
    Visualiza un DataFrame de Spark en Jupyter de forma similar a
Databricks display().

    Parámetros:
    - df: DataFrame de Spark
    - n: número máximo de filas a mostrar (default: 20)
    - columns: lista opcional de columnas a mostrar
    - sort_by: columna opcional para ordenar
    - ascending: orden ascendente o descendente
    """

    if columns:
        df = df.select(columns)

    if sort_by:
        df = df.orderBy(sort_by, ascending=ascending)

    pdf = df.limit(n).toPandas()
    display(pdf)
```