PROYECTO RA1:

Entregable 17 de Diciembre, GITHUB

Análisis y Procesamiento de Datos con Pandas, PySpark, ETL, Data Warehouse (SQLite) y Docker

---

1. Introducción

En este proyecto final pondrás en práctica todos los conocimientos adquiridos durante el curso. Trabajarás con un dataset real (asignado por el profesor) y construirás un flujo completo de análisis y procesamiento de datos, utilizando herramientas fundamentales del ecosistema de datos.

El proyecto es **individual** y tiene como objetivo simular un caso real básico de data analytics / data engineering.

---

2. Objetivos de aprendizaje

Al finalizar este proyecto, deberás demostrar que sabes:

✅ Explorar y limpiar datos con **Pandas**
✅ Procesar datos con **PySpark**
✅ Diseñar y construir **dos procesos ETL** (uno con Pandas y otro con PySpark)
✅ Cargar los resultados finales en **SQLite**
✅ Diseñar un **Data Warehouse (modelo dimensional)**
✅ Contenerizar el entorno del proyecto con **Docker**
✅ Documentar correctamente tu trabajo
✅ Organizar un proyecto con estructura profesional

---

3. Descripción general del proyecto

El proyecto se divide en dos flujos de trabajo paralelos:

🔹 Flujo 1: Pandas

1. Exploración y limpieza del dataset
2. Transformaciones con Pandas
3. Proceso ETL (Extracción, Transformación, Carga)
4. Creación de DDLs de las tablas del Datawarehouse
5. Carga final en **warehouse_pandas.db (SQLite)**

🔹 Flujo 2: PySpark

1. Procesamiento con Spark (carga del dataset limpio u original)
2. Transformaciones con Spark
3. Proceso ETL (Extracción, Transformación, Carga)
4. Creación de DDLs de las tablas del Datawarehouse
5. Carga final en **warehouse_pyspark.db (SQLite)**

🔹 Ambos flujos deben incluir:

- 1 tabla de hechos
- Mínimo 2 tablas de dimensiones
- Relaciones mediante claves

---

4. Requerimientos técnicos obligatorios

- Pandas
- PySpark

- Jupyter Notebook
- ETL en Pandas
- ETL en PySpark
- SQLite (2 bases de datos)
- Dockerfile y docker-compose.yml
- README.md
    - Modelo dimensional
    - Diagrama del flujo del proyecto
    - Ejemplos de preguntas y consultas a las tablas

---

5. Entregable

Debes entregar un enlace de Github con acceso al repositorio con toda la estructura del proyecto.

---

6. Estructura del proyecto

```
proyecto_nombre/
│
├── data/
│   └── dataset.csv (u otro formato)
│
├── notebooks/
│   ├── 01_pandas.ipynb
│   └── 02_pyspark.ipynb
│
│
├── warehouse/
│   ├── modelo_datawarehouse_pandas.sql
│   ├── modelo_datawarehouse_pyspark.sql
│   ├── warehouse_pandas.db
│   └── warehouse_pyspark.db
```

```
|
|Dockerfile
|docker-compose.yml
|
├── docs/
|   ├── README.md
|   └── diagrama.png (.drawio)
```

---

7. Instrucciones detalladas por fases

✅ Fase 1: Exploración y limpieza con Pandas (01_pandas.ipynb)

- Cargar el dataset
- Analizar tipos de datos
- Detectar y tratar valores faltantes / duplicados
- Normalizar / transformar columnas

✅ Fase 2: Procesamiento con PySpark (02_pyspark.ipynb)

- Crear SparkSession
- Cargar el dataset original o limpio
- Aplicar al menos 3 transformaciones (selección, filtrado, agregación, joins, nuevas columnas, etc.)
- Mostrar resultados

✅ Fase 3: Proceso ETL con Pandas (20%)

Debe incluir:

1. **Extracción:** lectura del dataset (original o limpio)
2. **Transformación:** limpieza + nuevas columnas / agregaciones
3. **Carga:** guardar resultado final en **SQLite (`warehouse_pandas.db`)** usando `to_sql`.

Debe incluir:

- 1 tabla de hechos
- Mínimo 2 tablas de dimensiones

✅ Fase 4: Proceso ETL con PySpark (20%)

Debe incluir:

1. **Extracción:** lectura del dataset
2. **Transformación:** filtrado, agrupaciones, joins, etc.
3. **Carga:** escribir el resultado final en **SQLite (`warehouse_pyspark.db`)**

Debe incluir:

- 1 tabla de hechos
- Mínimo 2 tablas de dimensiones

✅ Fase 5: Modelo de Data Warehouse (15%)

En la carpeta `warehouse/`:

Archivos requeridos:

- `modelo_datawarehouse_pandas.sql`

- `modelo_datawarehouse_pyspark.sql`

Cada uno debe contener:

- Definición de tablas (CREATE TABLE)
- Clave primaria en dimensiones
- Clave foránea en hecho
- Nombres descriptivos

✅ Fase 6: Docker (5%)

En `/docker/`:

- **Dockerfile obligatorio**
  Debe permitir:

- Instalar librerias de Python
- Ejecutar notebooks o scripts ETL

- Docker-compose.yml
- Dockerfile

✅ Fase 7: Documentación (10%)

`docs/README.md` debe incluir:

- Descripción del proyecto
- Explicación de cada fase
- Herramientas utilizadas
- Estructura de carpetas

- Instrucciones de ejecución (con y sin Docker)
- Explicación breve de cada ETL
- Cómo se cargaron los datos en SQLite
- Diagrama del modelo dimensional
- Consultas, y queries que se pueden realizar
- Conclusiones / aprendizajes

Diagrama:
Representar el flujo completo:
Dataset → Pandas → ETL → SQLite
Dataset → PySpark → ETL → SQLite

---

8. Organización / estructura / calidad del código (10%)

Se evaluará:

- Claridad de los notebooks
- Comentarios en el código
- Limpieza y orden en carpetas
- Nombres de variables adecuados
- Código legible

---

9. Rúbrica de evaluación (100%)

Criterio	Peso
Uso de Pandas	20%
Uso de PySpark	20%

ETL (Extracción, Transformación, Carga)	20%
Data Warehouse (modelo dimensional + SQLite)	15%
Documentación (README + diagrama)	10%
Organización / estructura / calidad del código	10%
Docker (contenedor funcional)	5%

✅ Total: 100%

---

10. Recomendaciones

✔ Trabaja por fases
✔ Guarda versiones intermedias
✔ Documenta mientras avanzas
✔ Prueba la carga en SQLite antes de terminar
✔ Haz el README al final cuando ya entiendas todo tu proceso
✔ Asegúrate de que el .zip abre correctamente y no falta nada

---

✅ Resultado esperado

Al finalizar, tendrás un proyecto **realista y completo** que demuestra tus habilidades en análisis y procesamiento de datos. Este proyecto puede formar parte de tu **portafolio profesional** para entrevistas o LinkedIn. La idea de subir tu solución a Github es usarlo como portafolio.

Ejemplo de algunos de los apartados del README esperado:

🔄 Explicación Detallada del Proceso ETL

1. Extracción (E)

Archivos origen:

- `notas_colegio.csv`

Acciones realizadas:

- Lectura de archivos CSV desde la carpeta `/data`.
- Detección y eliminación de valores faltantes (`NaN` o vacíos).
- Estandarización de nombres de columnas a `snake_case`.
- Conversión de fechas (`fecha`) al formato `YYYY-MM-DD`.
- Conversión de las notas a tipo `float`.

---

2. Transformación (T)

Objetivos:

- Unificar los datos de distintos colegios.
- Calcular métricas de rendimiento por estudiante.

Pasos realizados:

1. Unión de los datasets de ambos colegios con `concat`.
2. Eliminación de duplicados (por `id_estudiante`, `materia`, `fecha`).
3. Creación de nuevas columnas derivadas:
    - `nota_normalizada` = (nota / 10)
    - `estado` = 'Aprobado' si nota ≥ 5, 'Reprobado' si nota < 5
4. Separación de las tablas en componentes del modelo dimensional:
    - **dim_estudiante:** información única de cada estudiante.
    - **dim_materia:** catálogo de materias.
    - **dim_colegio:** información de los colegios.
    - **dim_fecha:** tabla de calendario generada automáticamente.
    - **fact_notas:** tabla de hechos con las relaciones y métricas (nota, nota_normalizada, estado).

---

3. Carga (L)

Se generó una base de datos **SQLite** llamada `datawarehouse.db`, y se cargaron las tablas mediante `SQLAlchemy`.

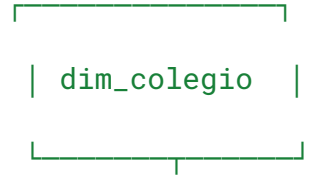
🧩 DDL de las Tablas (SQLite)

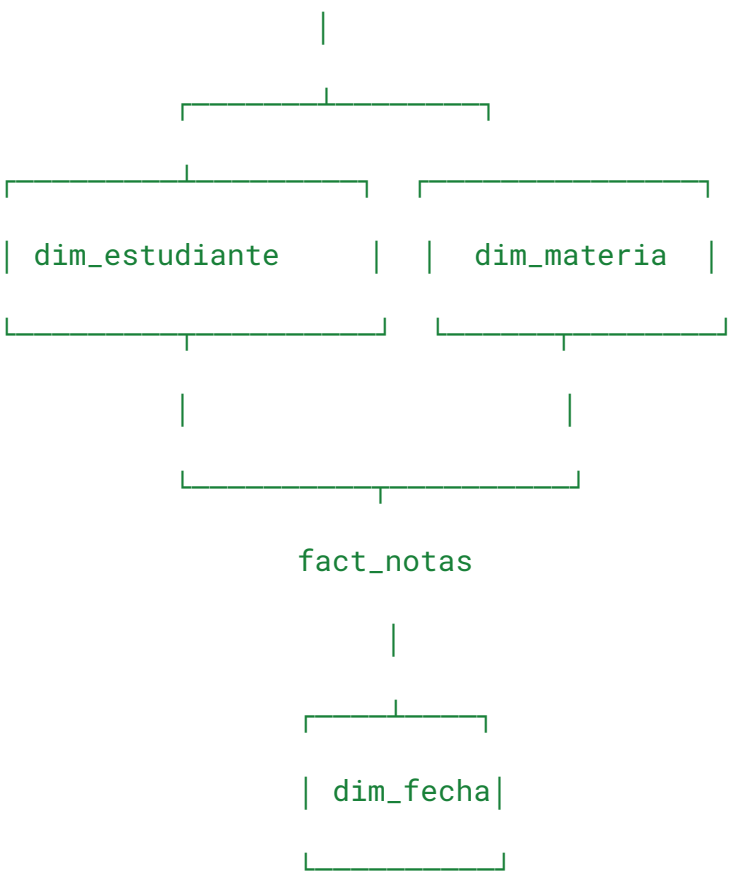
```
-- Tabla de estudiantes

CREATE TABLE dim_estudiante (

    id_estudiante INTEGER PRIMARY KEY,
```

```
    nombre TEXT NOT NULL
);

-- Tabla de materias
CREATE TABLE dim_materia (
    id_materia INTEGER PRIMARY KEY,
    nombre_materia TEXT NOT NULL
);

-- Tabla de colegios
CREATE TABLE dim_colegio (
    id_colegio INTEGER PRIMARY KEY,
    nombre_colegio TEXT NOT NULL
);

-- Tabla de fechas
CREATE TABLE dim_fecha (
    id_fecha INTEGER PRIMARY KEY,
    fecha DATE NOT NULL,
    anio INTEGER,
    mes INTEGER,
    dia INTEGER
);
```

```
-- Tabla de hechos
CREATE TABLE fact_notas (
    id_fact INTEGER PRIMARY KEY AUTOINCREMENT,
    id_estudiante INTEGER,
    id_materia INTEGER,
    id_colegio INTEGER,
    id_fecha INTEGER,
    nota REAL,
    nota_normalizada REAL,
    estado TEXT,
    FOREIGN KEY (id_estudiante) REFERENCES
dim_estudiante(id_estudiante),
    FOREIGN KEY (id_materia) REFERENCES dim_materia(id_materia),
    FOREIGN KEY (id_colegio) REFERENCES dim_colegio(id_colegio),
    FOREIGN KEY (id_fecha) REFERENCES dim_fecha(id_fecha)
);
```

🧠 Modelo Dimensional (Esquema en Estrella)





💡 Consultas y Preguntas de Análisis

Pregunta	Ejemplo de consulta
¿Cuál es la nota media por materia?	`SELECT m.nombre_materia, AVG(f.nota) FROM fact_notas f JOIN dim_materia m ON f.id_materia = m.id_materia GROUP BY m.nombre_materia;`
¿Qué porcentaje de aprobados tiene cada colegio?	`SELECT c.nombre_colegio, COUNT(CASE WHEN f.estado = 'Aprobado' THEN 1 END)*100.0 / COUNT(*) AS pct_aprobados FROM fact_notas f JOIN dim_colegio c`

	`ON f.id_colegio = c.id_colegio GROUP BY c.nombre_colegio;`
¿Cómo ha evolucionado el promedio de notas a lo largo del año?	`SELECT d.mes, AVG(f.nota) FROM fact_notas f JOIN dim_fecha d ON f.id_fecha = d.id_fecha GROUP BY d.mes;`