Resumen del ejercicio

Replicar un **sistema de ingeniería de datos** sencillo: los equipos recibirán dos ficheros CSV con patrón Clientes-YYYY-MM-DD.csv y Tarjetas-YYYY-MM-DD.csv. Dichos ficheros se deberán de dejar en una ruta (SFTP , unidad de red….) y los alumnos deben ser capaces de:

1. Obtener los ficheros (local, red o “fuente” simulada).
2. Leerlos y aplicar una **ETL** (limpieza, normalización, renombrado columnas, anonimización de datos sensibles).
3. Guardar los ficheros transformados.
4. Insertar los datos en una **base de datos** (SQL/Oracle/PostgreSQL/MySQL). Deben comprobar existencia de tablas: si no existen, crearlas; si existen, sólo insertar.
5. Automatizar el proceso (scripting, cron/CI/CD o pipeline simple).
6. Mantener todo el desarrollo en un **repositorio** (Git) con commits regulares y trabajo en ramas.
7. Documentar procedimiento, instrucciones de despliegue y pruebas.

Objetivos de aprendizaje

- Diseñar y construir un pipeline ETL básico en Python.
- Trabajar con ficheros CSV, manipulación de cadenas y normalización.
- Interactuar con una base de datos relacional (DDL + DML).
- Concebir y aplicar amonificación de datos sensibles.
- Practicar trabajo en equipo, git, y comunicación Scrum (representante).
- Automatizar y documentar.

Organización y dinámica de clase

- **Tamaño de grupos:** 5 alumnos por grupo, excepto 1 grupo de 6 (si el número de alumnos no divide, ajustar). Total, de miembros = variable.
- **Roles (a elegir por el equipo):** Product Owner (o representante / Scrum Master que será la interfaz contigo), Desarrollador ETL principal, Desarrollador DB / DBA, DevOps / Automatización, Tester / QA, Documentador (uno de los roles puede solaparse si son 5). El equipo decide asignaciones; **el representante** será la persona que hable conmigo y haga seguimiento.

- **Sesiones:** las clases presenciales/semanales se dedicarán a trabajo del proyecto. **Los primeros 45 minutos** de cada sesión serán exposición por grupos: avances, problemas detectados y soluciones, y tareas para la próxima semana. Cada representante y grupo dispondrá de 5–10 minutos para comentar lo dicho anteriormente. Pueden apoyarse con material visual si lo necesitan.
- **Comunicación:** cada representante debe enviar semanalmente (antes de la clase/exposición) un breve informe con: estado del proyecto, problemas, soluciones, entregables previstos.

Requisitos mínimos obligatorios (tecnologías)

- **Python** (versión 3+).
- Acceso a **repositorio Git** (GitHub/GitLab/Bitbucket o repositorio local) commit mínimo semanal y branching (feature branches).
- **Sistema de almacenamiento** legible/escribible por el pipeline (p. ej. carpeta compartida, S3 simulado, Azure Blob o storage local).
- **Base de datos relacional** levantada por los equipos (recomendado: PostgreSQL o Oracle). Se admite MySQL.
- Capacidad de **leer y escribir** en el almacenamiento.
- **Automatización**: script ejecutable (ej. python run_pipeline.py , crontab …) y programable mediante cron, GitHub Actions, o un scheduler simple.
- Uso de **librerías apropiadas** de Python (pandas, sqlalchemy, psycopg2/ cx_Oracle, etc. pueden usar las que consideren).
- Documentación en README.md con pasos para reproducir localmente y despliegue.

Tecnologías opcionales / libres

- Docker / docker-compose para levantar la DB.
- Airflow (o Prefect) para orquestación (opcional extra).
- Unit tests con pytest.
- CI con GitHub Actions para ejecutar tests y/o pipelines.
- Herramientas de logging (structlog / logging).

Entregables

1. Repositorio Git con código, README.md, requirements.txt (o pyproject), y scripts.
2. Pantallazo de ficheros transformados (CSV) en carpeta output/.
3. Scripts para crear tablas (si no existen) e insertar datos.
4. Documentación de decisiones (anonimización, normalizaciones).
5. Captura de la BD con datos insertados (puede ser dump.sql o capturas via psql).
6. Informes semanales (máx. 2 páginas) por grupo + presentación del proyecto en clase.
7. (Opcional) Pipeline automatizado (archivo cron, GitHub Action o docker-compose).
8. Deberán mostrar en proceso completo ante la clase con una exposición de la solución propuesta
9. Memoria completa, portada, miembros de grupo, funciones, diagrama de Gant, diagramas de flujo, soluciones aportadas, funcionamiento …

Criterios de evaluación (sobre 100)

- Funcionamiento ETL (lectura, limpieza, anonimizarían, guardado): **25**
- Inserción en BD y manejo de existencia de tablas: **15**
- Automatización y reproducibilidad (scripts, Docker/CI): **10**
- Calidad del código (estructura, modularidad, tests mínimos): **10**
- Documentación y README con pasos claros: **10**
- Trabajo en equipo y comunicación (informes + exposición de 45 min): **15**
- Extra (opcional, p. ej. orquestador, métricas, monitorización): **15**

Reglas exactas de los ficheros y patrón

- Nombres válidos: Clientes-YYYY-MM-DD.csv y Tarjetas-YYYY-MM-DD.csv. El pipeline solo **procesará** ficheros cuyos nombres cumplan exactamente ese patrón (fecha ISO YYYY-MM-DD), ignorando otros ficheros.
- Separador: punto y coma ;.

- Codificación: UTF-8 (si hay problemas, deberán detectarlo y probar otros encodings).
- Si hay otros ficheros, deben ignorarlos y dejar log que los han ignorado.

Estructura de los CSV (ejemplos y datos de muestra)

Clientes-2025-11-10.csv

Cod cliente;nombre;apellido1;apellido2;dni;correo;telefono

C001; María ;García ;Pérez;12345678A;maria.garcia@example.com; 612345678

C002;JOSÉ;LÓPEZ;HERNÁNDEZ;87654321B;jose.lopez@example.com;612345679

C003;Ana-María;Ruiz; ;23456789C;ana.ruiz@example.com;612345680

Tarjetas-2025-11-10.csv

cod_cliente;numero_tarjeta;fecha_exp;cvv

C001; 4532 1234 5678 9012 ;2026-09;123

C002;5500-0000-0000-0004;2027-03;321

C004;378282246310005;2025-12;999

Nota: cod_cliente es la clave que relacionará tablas.

Requisitos ETL detallados

1. **Detección y recuperación de ficheros:** listar el directorio fuente y filtrar para solo solo procesar **Clientes** y **Tarjetas**.
2. **Lectura robusta:** usar pandas.read_csv(..., sep=';', encoding='utf-8', dtype=str) con manejo de errores (bad lines / quoting).
    - Normalizar valores nulos
3. **Limpieza:**
    - Trim de espacios en todos los campos (str.strip()).
    - Normalizar mayúsculas/minúsculas donde corresponda: nombre capitalizado (con cuidado de guiones), correo en minúsculas.
    - Eliminar caracteres especiales innecesarios en campos alfanuméricos (por ejemplo en dni quitar espacios y guiones).
    - Eliminar acentos

- Validar telefono para dejar sólo dígitos (mantener prefijos si corresponde).
- Comprobar que el DNI introducido es correcto

4. **Renombrado columnas** a nombres consistentes en DB (eliminando espacios y caracteres especiales como son acentos). Documentarlo.
5. **Añadir dos columnas**
   - **DNI_OK**
     - **En caso de que el DNI sea correcto poner el valor Y en caso contrario N**
   - **DNI_KO**
     - **En caso de que el DNI sea incorrecto poner el valor Y en caso contrario N**
   - **Telefono_OK**
     - **En caso de que el Telefono sea correcto poner el valor Y en caso contrario N**
   - **Telefono_KO**
     - **En caso de que el Telefono sea incorrecto poner el valor Y en caso contrario N**
   - **Correo_OK**
     - **En caso de que el correo sea correcto poner el valor Y en caso contrario N**
   - **Correo_KO**
     - **En caso de que el correo sea incorrecto poner el valor Y en caso contrario N**
6. **Anonimización (datos sensibles):**
   - **Tarjetas:** nunca almacenar número completo en claro. (elegir opción y justificarla):
     - numero_tarjeta_masked: mostrar solo los 4 últimos dígitos y reemplazar resto por X (ejemplo: XXXX-XXXX-XXXX-9012).

        - numero_tarjeta_hash: hash irreversible (SHA-256) con **sal** por equipo (sal en variable de entorno o fichero no commitado). Ejemplo: hash = SHA256(salt + numero_tarjeta_sin_espacios).
        - Nunca guardar CVV en claro; eliminarlo o almacenar cvv_hash.
    - **DNI / identificadores personales**: si se considera sensible, aplicar hash o enmascarado parcial.
7. **Validaciones:** comprobar campos obligatorios (cod_cliente, correo mínimo 1) y registrar filas erróneas en errors/rows_rejected.csv(si registras el motivo de fallo, mejor).
8. **Salida:** generar ficheros output/Clientes-YYYY-MM-DD.cleaned.csv y output/Tarjetas-YYYY-MM-DD.cleaned.csv.
9. **Logging:** registrar pasos (inicio, fin, filas procesadas, filas insertadas, errores) en logs rotativos.