



CoE de Desarrollo
Democratización del Conocimiento



Kata Junior -Data Engineer| Integración y Cargue de Datos

Reto Junior – Data Engineer: Integración y Cargue de Datos

Contexto

El equipo de datos necesita integrar información que proviene de **múltiples fuentes heterogéneas** (archivos planos y motores de bases de datos). Actualmente, gran parte del trabajo diario incluye:

- **Consultas sobre bases de datos relacionales** (SQL, PL/SQL).
- **Procesos de ETL** con herramientas como **DataStage**. (Si conocen de datastage. Si no pueden usar otra herramienta)
- **Orquestación batch** con **Control-M**. (Si conocen de control m. Si no pueden usar otro orquestador)

Dado que los candidatos externos no tienen acceso a todas estas herramientas propietarias, se espera que puedan **simular el proceso con herramientas gratuitas**. Sin embargo, **si cuentan con un entorno de DataStage y desean usarlo, es totalmente válido** y se valorará positivamente.

Consideraciones de Seguridad en la parte baja del reto.


Objetivo del reto

Construir un flujo simple de **extracción, transformación y carga (ETL)** que:

1. **Extraiga datos** desde diferentes fuentes:
 - Una tabla de clientes en una base relacional (ejemplo: PostgreSQL o MySQL).
 - Un archivo CSV con transacciones financieras.
2. **Transforme los datos** aplicando:
 - Limpieza de registros incompletos o inválidos.
 - Normalización de fechas y montos.

- Enriquecimiento cruzando clientes ↔ transacciones.
3. **Cargue los resultados** en una nueva tabla consolidada llamada `movimientos_clientes`.
 4. **Automatice el flujo**: simular la ejecución batch con un script que ejecute paso a paso el ETL (en Python, Shell o Node.js).
-

Datos de Prueba

 Tabla `clientes` (BD relacional)

SQL

```
CREATE TABLE clientes (  
  cliente_id INT PRIMARY KEY,  
  nombre VARCHAR(100),  
  ciudad VARCHAR(50),  
  fecha_registro DATE  
);
```

```
INSERT INTO clientes VALUES  
(1, 'Ana Torres', 'Bogotá', '2024-03-01'),  
(2, 'Juan Pérez', 'Medellín', '2023-12-15'),  
(3, 'Carlos López', 'Cali', '2024-01-20');
```

yaml

 Archivo CSV `transacciones.csv`

```
id_transaccion,cliente_id,fecha,monto,tipo  
1001,1,2024-03-02,50000,COMPRA  
1002,2,2024-03-03,30000,ABONO  
1003,4,2024-03-01,20000,COMPRA ← Cliente no existe  
1004,3,2024-03-05,-10000,COMPRA ← Monto inválido
```

Herramientas Permitidas (Free/Open Source)

- **Bases de datos**: PostgreSQL, MySQL, SQLite.
- **ETL/Transformación**: Python (Pandas), Talend Open Studio, Pentaho Data Integration (Kettle).
- **Orquestación**: Cron jobs, Airflow local, o un script Bash/Batch que simule la ejecución (en vez de Control-M).

- **Versionamiento:** GitHub/GitLab repositorio público.
 - **Opcional: IBM DataStage** 🖐️ si el candidato cuenta con acceso a un entorno válido, puede usarlo para construir el flujo ETL.
-

Entregables

1. Script SQL para la creación de tablas y carga inicial de clientes.
 2. Archivo CSV con transacciones.
 3. Script de transformación (Python/Node.js/Shell o flujo DataStage si aplica) que:
 - Valide clientes.
 - Filtre transacciones inválidas.
 - Genere archivo `movimientos_clientes.csv`.
 4. Script de orquestación simple (ej: `run_etl.sh`) que ejecute todo el flujo paso a paso.
 5. Repositorio GitHub con el código, scripts y datos de prueba.
 6. Exposición (5 minutos): explicar el flujo, las transformaciones y cómo se automatizó el proceso.
-

Criterios de Evaluación

- Dominio de **SQL** (consultas y manipulación de datos).
 - Capacidad para **integrar datos de distintas fuentes**.
 - Correcto manejo de **validaciones y calidad de datos**.
 - Creatividad en la **simulación de orquestación batch**.
 - Claridad en la explicación técnica.
 - Uso de **DataStage (si está disponible)** como diferenciador positivo.
-

Consideraciones de Seguridad

- No subir credenciales reales ni secretos a repositorios públicos.
 - Usar datos simulados en todos los ejemplos.
 - Versionar el código en repositorios personales.
 - Se recomienda el uso de `.gitignore` para excluir archivos sensibles.
-

¡Importante!

📢 **¡Atención, equipo!** 🚀 . General

Queremos recordarles algo **MUY IMPORTANTE** para mantener la seguridad de nuestra información y la del banco. Recientemente, tuvimos un incidente porque en una **Kata de desarrollo** se usó un repositorio real de la organización y accidentalmente se expuso un **Secret Key** en GitHub. 😱

Esto representa un **riesgo grave** para la seguridad, por lo que necesitamos su ayuda para evitarlo en el futuro.

📌 **Recomendaciones clave:**

- ♦️ **🚫 No usen repositorios reales del banco en Katas.**
 - Las Katas son para práctica, ¡usen repositorios personales, no usar sandbox del banco pueden usar sandbox personales!
- ♦️ **🔒 Nunca suban credenciales, keys o información sensible.**
 - Si necesitan datos de prueba, usen valores ficticios (ej: `"token_ejemplo_123"`).
- ♦️ **✅ Verifiquen antes de hacer commit/push.**
 - Usen herramientas como `git-secrets` o `.gitignore` para evitar subir archivos críticos.
- ♦️ **🛡️ Si ven algo raro, repórtenlo de inmediato.**

💡 **Alternativas seguras para Katas:**

- **GitHub Personal:** Crear un repo en sus cuentas personales (no vinculado al banco).
- **Entornos de prueba:** Usar bases de datos o APIs mock (como Mockaroo, JSON-Server).
- **Datos falsos:** Siempre que sea posible, generen datos simulados.

☀️ **¡Juntos protegemos nuestra seguridad!** ☀️

Este tipo de errores pueden tener consecuencias serias, pero con buenas prácticas los evitamos. ¡Gracias por su colaboración!

¡Éxitos con el Reto! 🚀

El acompañamiento

- Democratización del conocimiento