☑ 3. ETL / ELT — 50 preguntas

1. Explica la diferencia entre ETL y ELT.

- **ETL (Extract, Transform, Load):** Primero extrae los datos, luego los transforma y finalmente los carga en el destino.
    - **Ejemplo:** Extraer datos de un ERP, limpiar y normalizar en un servidor intermedio, y luego cargarlos a un Data Warehouse.
- **ELT (Extract, Load, Transform):** Primero se extraen y cargan los datos en el destino y la transformación se hace ahí.
    - **Ejemplo:** Extraer datos de logs, cargarlos a un Data Lake en la nube y transformarlos usando SQL o Spark directamente allí.

2. ¿Cuándo preferir ETL sobre ELT?

- ETL es preferible cuando el **sistema destino no tiene capacidad de transformación** o cuando se quiere **controlar calidad antes de cargar**.
- Ejemplo: Bases de datos tradicionales con poca potencia de cómputo.

3. ¿Qué retos presenta la transformación en el origen versus en el destino?

- **Origen:** Puede afectar sistemas productivos, requiere coordinación con sistemas fuente.
- **Destino:** Se necesita mayor capacidad de cómputo, pero permite mayor flexibilidad y menor impacto en producción.

4. Describe un proceso estándar de extracción de datos.

- Pasos típicos:
    1. Conexión al origen.
    2. Selección de datos (query o API).
    3. Filtrado y validación mínima.
    4. Exportación a formato intermedio (CSV, JSON, parquet).
- Ejemplo: Extraer ventas del ERP cada día, filtrando solo datos del último mes.

5. Menciona buenas prácticas para limpiar datos.

- Normalizar formatos (fechas, monedas).
- Eliminar duplicados.
- Manejar valores nulos o inválidos.
- Validar consistencia referencial.
- Ejemplo: Transformar “01/12/23” y “2023-12-01” a un formato estándar ISO (`YYYY-MM-DD`).

6. ¿Qué es un CDC y qué técnicas existen?

- **CDC (Change Data Capture):** Detecta cambios en la fuente y los replica al destino.
- Técnicas:
    - Basado en **timestamp**.
    - Basado en **logs de transacción**.
    - Basado en **triggers** en la base de datos.

7. ¿Qué problemas soluciona un proceso de upsert?

- Upsert = **Update + Insert**, permite insertar registros nuevos y actualizar existentes, evitando duplicados.
- Ejemplo: Al cargar clientes, si ya existe el ID, se actualiza el email; si no, se inserta.

8. ¿Cómo evitar cargas duplicadas?

- Usar **llaves primarias únicas**.
- Implementar **hash de fila** para comparar cambios.
- Realizar **control de checkpoint** para retomar procesos.
- Ejemplo: Hash sobre `ID + fecha` para detectar si la fila ya se cargó.

9. ¿Cómo manejar errores en procesos de ingesta?

- Implementar **reintentos automáticos** (retry policy).
- Guardar errores en un **log de fallos**.
- Validar datos antes de insertarlos.
- Ejemplo: Si falla una fila por formato incorrecto, moverla a una tabla de errores y continuar con las demás.

10. ¿Qué patrón usarías para cargas masivas diarias?

- **Batch processing**, con partición de datos y control de checkpoint.
- Ejemplo: Cargar cada noche todas las transacciones del día anterior en bloques de 1 millón de registros.

11. ¿Qué es un staging layer?

- Es una **capa intermedia** donde los datos se cargan antes de ser transformados o movidos al destino final.
- Propósito: Validar, limpiar y preparar datos.
- **Ejemplo:** Extraer archivos CSV del ERP, cargarlos en tablas temporales (`staging_customers`) antes de insertarlos al Data Warehouse.

12. Explica el concepto de landing zone.

- Es el **primer punto de llegada de datos** sin procesar en el sistema ETL.
- Los datos en la landing zone son **crudos**, sin transformación ni limpieza.

- **Ejemplo:** Archivos CSV de ventas diarios se guardan tal cual en un bucket de S3 antes de procesarlos.

13. ¿Qué herramientas ETL has usado y cuál prefieres?

- Herramientas populares: **Informatica, Talend, Apache Nifi, SSIS, Airflow, dbt**.
- Preferencia depende de la necesidad:
    - Airflow para **orquestación de pipelines**.
    - dbt para **transformaciones en el destino (ELT)**.
    - Ejemplo: Airflow se puede usar para disparar un flujo que primero extrae de un ERP, luego transforma en staging, y finalmente carga al Data Warehouse.

14. ¿Qué es un merge incremental?

- Es una estrategia de ETL donde **solo se procesan los datos nuevos o modificados** desde la última carga.
- Reduce tiempo y recursos en comparación con cargas completas.
- **Ejemplo:** Procesar solo las órdenes que cambiaron desde la última ejecución usando una columna `last_updated`.

15. Explica estrategias de re-procesamiento.

- **Re-procesamiento completo:** Recargar toda la fuente desde cero (costoso pero garantiza consistencia).
- **Re-procesamiento incremental:** Solo los datos faltantes o modificados (eficiente).
- **Ejemplo:** Si un pipeline falla por un archivo corrupto, se puede re-procesar solo ese archivo o lote.

16. ¿Cómo garantizar idempotencia en un pipeline?

- Idempotencia: Ejecutar varias veces el pipeline **no cambia el resultado final**.
- Estrategias:
    - Usar llaves únicas.
    - Upserts en lugar de inserts puros.
    - Control de marcas de tiempo.
- **Ejemplo:** Ejecutar la carga diaria de clientes dos veces no genera duplicados gracias al `MERGE` sobre `customer_id`.

17. ¿Qué problemas genera el late arriving data?

- Datos que llegan **después del tiempo esperado**.
- Problemas: inconsistencia en reportes, métricas erróneas, necesidad de recalcular agregaciones.
- **Ejemplo:** Transacciones del 1 de diciembre que llegan el 3 de diciembre y afectan el KPI de ventas diario.

18. ¿Cómo manejar registros corruptos?

- Validación en staging o landing zone: detectar errores de formato, tipos o valores nulos.
- Mover registros corruptos a una tabla de errores (`error_table`) para revisión.
- Notificar a responsables para corrección.
- **Ejemplo:** Un CSV con fecha `32/13/2023` se mueve a la tabla de errores y se registra en logs.

19. Explica batch processing vs stream processing.

- **Batch:** Procesa datos en bloques o lotes a intervalos regulares.
  - Ejemplo: Cargar ventas diarias en un Data Warehouse cada noche.
- **Stream:** Procesa datos en tiempo real o casi real.
  - Ejemplo: Actualizar dashboards de tráfico web minuto a minuto usando Kafka y Spark Streaming.

20. ¿Cómo definir tamaños adecuados de batch?

- Depende de: capacidad del destino, volumen de datos, tiempo de ventana disponible.
- Estrategias:
  - Probar distintos tamaños para optimizar throughput y minimizar fallas.
  - Dividir en lotes manejables (p.ej., 10k–1M filas).
- **Ejemplo:** Para un Data Warehouse en la nube, batches de 100k filas permiten un balance entre velocidad y consumo de recursos.

21. ¿Qué métricas usas para monitorear procesos ETL?

- **Tiempo de ejecución:** cuánto tarda cada job o pipeline.
- **Throughput:** volumen de datos procesados por unidad de tiempo.
- **Errores/fallos:** número de registros fallidos o rechazados.
- **Alertas:** notificaciones ante retrasos o errores.
- **Ejemplo:** Monitorear un pipeline que procesa 1M de registros diarios; si tarda más de 2 horas, disparar alerta.

22. ¿Cómo optimizar la velocidad de carga?

- Usar **cargas en paralelo**.
- Aplicar **transformación pushdown** al origen o destino.
- Ajustar **batch sizes** y particiones.
- **Ejemplo:** Cargar tablas de ventas por partición de fecha en lugar de todo el histórico a la vez.

23. ¿Qué es el schema drift y cómo manejarlo?

- Schema drift: cambios inesperados en la estructura de los datos de origen.
- Manejo:
    - Validación de esquema antes de cargar.
    - Metadata-driven ETL que se ajusta automáticamente.
    - Alertas en caso de cambios.
- **Ejemplo:** Una columna `phone_number` que cambia de tipo `integer` a `string`. El pipeline detecta y adapta la carga.

24. ¿Cómo manejar cambios en la estructura de origen?

- Estrategias:
    - Agregar columnas nuevas sin romper el pipeline.
    - Mapear columnas renombradas.
    - Usar staging para revisar cambios antes de transformar.
- **Ejemplo:** La columna `customer_address` se divide en `street` y `city`; se ajusta el pipeline para aceptar ambas nuevas columnas.

25. ¿Qué es data enrichment?

- Proceso de **agregar información adicional** a los datos para hacerlos más valiosos.
- Ejemplo: Añadir datos de ubicación geográfica a registros de clientes usando su código postal.

26. ¿Cómo validar que una carga finalizó correctamente?

- Verificar número de registros cargados vs origen.
- Revisar logs de errores.
- Confirmar integridad referencial en destino.
- **Ejemplo:** Extraer 1M de filas de ventas y asegurar que 1M se cargaron en la tabla destino sin errores.

27. ¿Qué es el checkpointing?

- Técnica que guarda el **estado intermedio** del proceso ETL para poder reiniciar desde ahí en caso de fallo.
- Evita reprocesar datos ya procesados.
- **Ejemplo:** En un flujo de 10 archivos diarios, marcar que se procesaron los 7 primeros; si falla el 8º, reiniciar desde ahí.

28. ¿Cómo manejar procesos ETL dependientes entre sí?

- Usar **orquestadores** (Airflow, Luigi).
- Definir dependencias entre jobs (DAG: Directed Acyclic Graph).
- Implementar **checks de finalización** antes de disparar el siguiente proceso.
- **Ejemplo:** No cargar datos de facturas hasta que se hayan cargado los datos de clientes.

29. Explica qué es una tabla staging.

- Tabla temporal donde se cargan **datos crudos o parcialmente procesados**.
- Propósito: Validar y transformar antes de la carga final.
- **Ejemplo:** `staging_orders` recibe todas las órdenes del ERP, se limpia y luego se inserta en `dw_orders`.

30. ¿Qué es un orchestrator?

- Herramienta que **coordina la ejecución de pipelines ETL**, manejando dependencias, retries y alertas.
- Ejemplo: **Airflow** permite ejecutar un flujo: extraer -> transformar -> cargar, asegurando que cada paso depende del anterior y notificando en caso de error.

31. ¿Cómo garantizar calidad de datos durante la transformación?

- Validaciones de integridad: llaves primarias, tipos de datos, relaciones.
- Reglas de negocio: valores esperados, rangos, consistencia.
- Pruebas automatizadas y alertas en errores.
- **Ejemplo:** Antes de cargar ventas al Data Warehouse, validar que `total_amount >= 0` y que `customer_id` exista en la tabla de clientes.

32. ¿Qué estrategias de logging implementas?

- **Logs de auditoría:** registros de cuándo y qué datos se procesaron.
- **Logs de errores:** filas rechazadas y motivos.
- **Logs de performance:** tiempos de ejecución, throughput.
- **Ejemplo:** Registrar en un log que 1M de filas se cargaron en 45 min, con 5 filas con errores de formato.

33. ¿Cómo manejar fallas transitorias?

- Reintentos automáticos (retry policy).
- Backoff exponencial para evitar saturar sistemas.
- Monitorización para alertas si el problema persiste.
- **Ejemplo:** Fallo al conectar con el API del ERP; reintentar cada 5 minutos hasta 3 veces antes de notificar.

34. ¿Qué es un retry policy?

- Política que define **cuándo y cuántas veces volver a intentar** un proceso ETL que falla.
- Puede incluir: número máximo de reintentos, intervalos y backoff exponencial.
- **Ejemplo:** Reintentar carga de archivos hasta 5 veces con intervalos de 2, 4, 8, 16, 32 minutos.

35. Explica horizontal scaling en procesos ETL.

- Aumentar **número de nodos o máquinas** para procesar más datos en paralelo.
- Ideal para Big Data o cargas distribuidas.
- **Ejemplo:** Usar 10 nodos Spark para procesar 1TB de logs en paralelo.

36. Explica vertical scaling en procesos ETL.

- Aumentar **capacidad de una máquina** (CPU, memoria, disco) para procesar más datos.
- Menos flexible que horizontal scaling, pero útil si el pipeline es monolítico.
- **Ejemplo:** Incrementar memoria de un servidor ETL de 16GB a 64GB para acelerar cargas de grandes archivos CSV.

37. ¿Cómo manejar archivos grandes (100GB+)?

- Dividir en **chunks o particiones**.
- Procesamiento **streaming** o por lotes.
- Comprimir o usar formatos columnar (Parquet, ORC).
- **Ejemplo:** Leer un archivo CSV de 200GB en bloques de 1GB y cargar cada bloque en paralelo.

38. ¿Cómo paralelizar cargas?

- Dividir datos por **partición lógica** (fecha, región, ID).
- Ejecutar múltiples procesos ETL simultáneamente.
- **Ejemplo:** Cargar datos de clientes por región: cada región en un hilo diferente para acelerar la ingesta.

39. ¿Cómo optimizar un pipeline saturado?

- Identificar **cuellos de botella** (origen, transformación o destino).
- Ajustar batch sizes, paralelismo o indexing en la base de datos.
- Implementar **caching o pushdown de transformaciones**.
- **Ejemplo:** Un pipeline lento al insertar millones de filas, se optimiza usando `bulk insert` y transformaciones en la base destino.

40. ¿Qué herramientas recomiendas para un entorno cloud?

- **ETL/ELT:** AWS Glue, Azure Data Factory, Google Cloud Dataflow.
- **Orquestación:** Airflow, Prefect.
- **Almacenamiento:** S3, BigQuery, Snowflake, Redshift.
- **Ejemplo:** Extraer datos desde un ERP local y cargar en Snowflake usando AWS Glue para transformaciones y Airflow para orquestar el flujo.

41. ¿Qué es data deduplication?

- Proceso de **eliminar registros duplicados** para garantizar consistencia y optimizar almacenamiento.
- Estrategias: comparación de llaves únicas o hash de filas.
- **Ejemplo:** Dos registros de cliente con el mismo `customer_id` y nombre, se consolidan en uno solo antes de cargar al Data Warehouse.

42. ¿Qué problemas trae out-of-order events?

- Datos que llegan **fuera de secuencia temporal** pueden generar métricas incorrectas o inconsistencias en agregaciones.
- Solución: reordenar por timestamp o usar buffers temporales.
- **Ejemplo:** Una transacción con hora `10:05` llega después de otra con hora `10:10`; al calcular ventas acumuladas, se corrige el orden por timestamp.

43. ¿Qué sistemas de mensajes usarías para stream ETL?

- Kafka, RabbitMQ, Kinesis, Pub/Sub.
- Permiten **procesamiento en tiempo real** y escalabilidad.
- **Ejemplo:** Enviar logs de aplicación a Kafka y procesarlos en Spark Streaming para dashboards en tiempo real.

44. Explica exactly-once processing.

- Garantiza que **cada evento se procesa exactamente una vez**, evitando duplicados o pérdidas.
- Estrategias: idempotencia, transacciones, checkpointing.
- **Ejemplo:** Al procesar pagos en tiempo real, cada pago se contabiliza una sola vez, incluso si falla un reintento.

45. ¿Cómo manejar incompatibilidad de tipos en la transformación?

- Validar y **convertir tipos** antes de cargar.
- Implementar reglas de fallback o nulos para valores inválidos.
- **Ejemplo:** Convertir `string` a `date` usando un formato estándar; si falla, guardar en tabla de errores.

46. ¿Qué es un workflow DAG?

- DAG = Directed Acyclic Graph. Representa **dependencias entre tareas sin ciclos**.
- Permite orquestar pipelines ETL secuenciales o paralelos.
- **Ejemplo:** En Airflow, extraer datos -> limpiar -> transformar -> cargar; cada tarea depende de la anterior, formando un DAG.

47. ¿Cómo probar un pipeline ETL?

- Pruebas unitarias de transformaciones.
- Test de integración para conexiones y cargas.
- Pruebas de performance y volumen de datos.
- **Ejemplo:** Validar que una función que calcula `total_amount` sobre ventas devuelva resultados correctos antes de ejecutar todo el pipeline.

48. ¿Qué es metadata-driven ETL?

- ETL que **usa metadatos para definir transformaciones y cargas** dinámicamente.
- Facilita cambios en origen, destino o reglas de negocio sin modificar código.
- **Ejemplo:** Una tabla de metadatos define qué columnas cargar y transformar; si cambia el origen, el pipeline se adapta automáticamente.

49. ¿Qué es la "transformación pushdown"?

- Ejecutar transformaciones **en la base de datos o motor de destino/origen** en lugar de extraer datos y transformarlos externamente.
- Mejora rendimiento y reduce tráfico.
- **Ejemplo:** Filtrar y agregar ventas directamente en SQL en el Data Warehouse antes de cargar el resultado al pipeline.

50. ¿Cómo asegurar que un pipeline es escalable?

- Diseñar procesos **paralelizables** y tolerantes a fallos.
- Usar arquitecturas **distribuidas** y escalables horizontalmente.
- Monitorizar y optimizar recursos según crecimiento de datos.
- **Ejemplo:** Un pipeline de logs que puede procesar 1M de eventos por hora y, al crecer a 10M, se escala añadiendo nodos Spark y particionando los datos por fecha.