**Gerencia Servicios de Nómina a Empleados**

**Prueba Técnica Analista III**

**Shamuel Steven Molina Duque**

**Caso practico**

1. En la Gerencia Servicios de Nómina a Empleados, existe un proceso mensual que consolida información desde múltiples fuentes:

* Una base de datos en SQL con tablas de contratos, novedades, horas extras y deducciones.
* Archivos Excel externos con información de bonificaciones variables y ausentismos.
* Un modelo en Power BI que resume y visualiza el resultado para la VP Talento y Cultura y el Entorno al Serv de Func Corporativas.
* Unos scripts en Python que realizan validaciones de reglas de negocio (ej. topes de bonificación, cálculos proporcionales por ausencias, validación de duplicados).

El proceso actual tarda 3 horas, presenta errores recurrentes en los reportes y genera reprocesos cada cierre de mes. Tu objetivo es rediseñar este proceso para hacerlo más robusto, auditable garantizando la integridad y exactitud.

* ¿Cómo realizarías el diagnóstico técnico y funcional del proceso actual?

Para realizar un diagnóstico técnico del proceso actual, empezaría por mapear cada fuente involucrada (SQL, Excel, Python, Power BI) y evaluar los tiempos de ejecución por etapa: carga, transformación, validación y visualización. Identificaría posibles cuellos de botella, como lecturas lentas desde Excel, cálculos poco eficientes en Python, consultas SQL sin índices adecuados o modelos de Power BI mal estructurados. Además, revisaría la calidad del código para eliminar redundancias, subconsultas innecesarias o falta de paralelismo que pueda estar ralentizando el flujo general.

Y desde el enfoque funcional, me enfocaría en entender qué áreas consumen cada componente del proceso y qué tipo de errores se presentan con mayor frecuencia (sean duplicados o cálculos incorrectos). Para ello, lo revisaría con usuarios clave de talento humano, TI y nómina, con el fin de documentar con claridad las reglas de negocio actuales, ítems como topes de bonificación, proporciones por ausencias y condiciones especiales. Esto más que para optimizar el flujo, es para asegurar que cada parte del proceso responde correctamente a sus objetivos.

* ¿Qué mejoras aplicarías sobre las consultas SQL y los scripts en Python?

Para mejorar las consultas SQL dentro del proceso, comenzaría por asegurar que todas las tablas involucradas cuenten con índices en las columnas utilizadas en las cláusulas JOIN y WHERE, lo que permite acelerar significativamente los tiempos de respuesta. También reemplazaría subconsultas anidadas por expresiones comunes de tabla (WITH o CTEs), mejorando tanto la legibilidad como el rendimiento, también evitar el uso innecesario de DISTINCT, ya que puede ocultar problemas de duplicidad y afectar la eficiencia. En caso de trabajar sobre un data warehouse (landing zone), verificaría el uso adecuado del particionamiento para aprovecharlo al máximo en los filtros.

En cuanto a los scripts en Python, propondría refactorizarlos en funciones modulares reutilizables para facilitar el mantenimiento y la escalabilidad. Implementaría validaciones de reglas de negocio utilizando bibliotecas como pandas y pydantic, asegurando integridad en los datos antes de su carga. Además, incluiría trazabilidad mediante el módulo logging, permitiendo registrar errores y eventos clave del proceso. Finalmente, optimizaría el procesamiento evitando bucles innecesarios y priorizando operaciones vectorizadas, lo que mejora el rendimiento en el manejo de grandes volúmenes de datos. Todo lo anterior bajo los manuales de buenas prácticas.

* ¿Qué estrategia ETL propondrías para integrar todas las fuentes de forma más eficiente y automatizada?

Para lograr una integración eficiente y completamente automatizada de las fuentes de datos, propondría una estrategia ETL que opere de forma centralizada sobre la Landing Zone, asegurando que el 100% de los insumos provengan de esta capa y eliminando dependencias externas. El desarrollo se orquestaría desde Orquestador 2.0, permitiendo una calendarización robusta y controlada de cada etapa del flujo.

En la fase de extracción, se estructurarían consultas SQL eficientes directamente sobre la LZ, asegurando trazabilidad, cumplimiento de políticas internas y consistencia en la entrada de datos. Las transformaciones se implementarían mediante scripts Python y SQL que han superado los pipelines de validación técnica y de estilo, lo que garantiza la calidad y mantenibilidad del código. Todo el desarrollo se gestionaría a través de un repositorio en Azure DevOps, respetando las convenciones establecidas para el nombramiento, versionamiento y calendarización.

La capa de carga consolidaría la información en un DataMart de Nómina limpio y validado, expuesto para su consumo en herramientas analíticas como Power BI. El proceso completo se ejecutaría dentro de DAGs controlados, con trazabilidad paso a paso, monitoreo de eventos y alertas configuradas ante cualquier fallo técnico o de validación lógica, soportado por un esquema de trabajo colaborativo con roles diferenciados y claramente definidos en desarrollo, QA y aprobación.

Además, se cumplirían los estándares organizacionales en cuanto a documentación técnica y funcional, levantamiento de requisitos, diseño de malla y tablas de resultados. Todo esto respaldado por mecanismos de soporte operativo ante incidentes, lo cual permite una ejecución confiable, auditable y escalable en el tiempo.

* ¿Cómo rediseñarías el modelo de Power BI para hacerlo más dinámico, auditable y funcional?

Para rediseñar el modelo en Power BI de forma más dinámica, auditable y funcional, estructuraría los datos bajo un modelo estrella, donde la tabla de hechos contenga los movimientos de nómina y se relacionen con dimensiones bien definidas como empleados, conceptos y fechas. Las relaciones serían de uno a muchos y sin ambigüedades, lo cual facilita el análisis y mejora el rendimiento del modelo. Las medidas en DAX estarían claramente definidas con nombres estandarizados, y todas las validaciones de reglas de negocio se aplicarían en la etapa de ETL, evitando sobrecargar el modelo con lógica compleja.

Desde el enfoque de auditoría y usabilidad, incluiría una pestaña de “Control” que muestre información clave como la fecha de última actualización, el número de registros procesados y errores detectados. También implementaría visualizaciones que permitan rastrear el origen de cada bonificación o novedad. En cuanto a la experiencia del usuario, diseñaría paneles adaptados a cada perfil (VP, analistas, HRBP), con filtros claros por mes, unidad o tipo de novedad, asegurando que cada usuario acceda rápida y fácilmente a la información relevante para su rol.

* ¿Qué controles, logs o reportes implementarías para asegurar trazabilidad, control de errores y cumplimiento de políticas de nómina?

Para asegurar la trazabilidad y el cumplimiento de políticas en el proceso de nómina, implementaría controles que validen la estructura y el esquema de los archivos Excel recibidos, verificando el número de columnas, encabezados esperados y consistencia del contenido. También incluiría chequeos automáticos de duplicados y validaciones de topes máximos y mínimos por tipo de concepto, garantizando que los datos cumplan con las reglas de negocio antes de ser procesados.

En paralelo, establecería un sistema de logs, ya sea en tablas o archivos planos, que registre la fecha y hora de ejecución, la fuente procesada, la cantidad de registros leídos, cuántos fueron válidos o inválidos y los errores detectados junto con su causa. Estos logs alimentarían reportes automáticos de validación enviados por correo tras cada cierre mensual, y una bitácora completa quedaría disponible para procesos de auditoría. Como parte de las políticas de gobernanza, bloquearía registros que no cumplan las reglas y habilitaría un sistema de almacenamiento seguro y versionado para los archivos Excel, evitando pérdidas o sobreescritura accidental de la información.

1. Dada la siguiente consulta, ¿Cómo la optimizarías?

SELECT c.Region, AVG(t.Saldo) AS Saldo\_Promedio

FROM Clientes c

JOIN Cuentas ct ON c.ID\_Cliente = ct.ID\_Cliente

JOIN Transacciones t ON ct.ID\_Cuenta = t.ID\_Cuenta

WHERE t.Fecha BETWEEN '2024-01-01' AND '2024-12-31'

GROUP BY c.Region;

Esa consulta la optimizaría así (y te cuento abajo por qué de esa forma):

WITH transacciones\_filtradas AS (

SELECT ID\_Cuenta, Saldo

FROM Transacciones

WHERE Fecha >= '2024-01-01' AND Fecha <= '2024-12-31'

)

SELECT cli.Region, AVG(tf.Saldo) AS Saldo\_Promedio

FROM Clientes cli

JOIN Cuentas cta ON cli.ID\_Cliente = cta.ID\_Cliente

JOIN transacciones\_filtradas tf ON cta.ID\_Cuenta = tf.ID\_Cuenta

GROUP BY cli.Region;

Esta forma de optimización es la que encuentro más ideal porque reduce significativamente el volumen de datos procesados al aplicar los filtros de fecha antes de realizar los JOINs, lo que mejora el rendimiento de la consulta en motores como Impala. Además, al estructurar la lógica en una subconsulta (WITH), se facilita la escalabilidad, mantenimiento y reutilización en otros procesos y también que la estructuré siguiendo el manual de buenas practicas de la LZ.

1. ¿Como llevarías a producción una ETL bajo el modelo de calendarización 2.0? describe los pasos

Para llevar una ETL a producción bajo el modelo de Calendarización 2.0, iniciaría construyendo la rutina desde el package generator, asegurando que la estructura base y la configuración sean compatibles con Orquestador 2.0. Desde el primer momento, el desarrollo seguiría las buenas prácticas de codificación, tanto en Python como en SQL, priorizando un diseño modular, parametrizable y portable, siempre operando exclusivamente sobre insumos disponibles en la LZ.

Durante el desarrollo, se mantendría una documentación clara y actualizada del flujo, incorporando reglas de negocio, dependencias y resultados esperados. Una vez completada la construcción, realizaría las ejecuciones necesarias para generar los logs de compilación requeridos según el tipo de despliegue, asegurando trazabilidad desde el entorno local hasta la integración en ambientes superiores.

Posteriormente, se levantaría el pull request con la plantilla oficial, detallando validaciones técnicas, pruebas ejecutadas y cumplimiento de estándares. Este proceso activa los pipelines automáticos de revisión, que validan tanto la calidad del código como la completitud de los registros de pruebas. Una vez aprobado, el artefacto se distribuye progresivamente por los entornos, validando funcionalidad y cumplimiento técnico en cada fase.

Al llegar a producción, la rutina se incorpora a la malla operativa con su configuración de frecuencia y dependencias establecidas. Desde allí, queda bajo el monitoreo continuo de un equipo asignado, con capacidad de escalar cualquier incidente operativo, garantizando estabilidad, trazabilidad y mantenimiento ágil bajo los lineamientos del modelo vigente.

**Prueba técnica (Todos los archivos los adjunté en el correo)**

El siguiente enunciado pretende evaluar sus conocimientos en ingeniería de datos, lenguajes de programación y estructuración de soluciones técnicas.

Eres parte del equipo de Evolución de la Gerencia de Nomina. Debes desarrollar un proceso que consolide la información de liquidación mensual de empleados, aplicando reglas de negocio específicas, integrando datos de distintas fuentes y generando un reporte final limpio y validado.

1. Integración de datos:

* Une la información de empleados, horas extras, deducciones y bonificaciones.
* Calcula el salario total líquido por empleado del mes (junio 2024), teniendo en cuenta:
* Horas extras diurnas: pagan al 25% + del valor hora ordinaria.
* Horas extras nocturnas: pagan al 75% + del valor hora ordinaria.
* Bonificaciones deben sumarse al salario.
* Deducciones deben restarse del total.

1. Reglas de negocio:

* Solo procesar empleados activos (fecha\_ingreso anterior al 2024-06-01).
* El salario no debe superar 10 millones después de aplicar bonificaciones, si lo hace generar alerta.
* Si un empleado tiene más de $700.000 en deducciones, marcarlo como "Alerta financiera".

3. Automatización:

* Usa Librerías necesarias para cargar y procesar los datos.
* Usar el Motor de bases de datos que prefieras

4. Genera un archivo Excel final con el siguiente formato:

| **id\_empleado** | **nombre** | **salario\_basico** | **total\_horas\_extras** | **total\_bonificaciones** | **total\_deducciones** | **salario\_liquido** | **alerta\_financiera** |
| --- | --- | --- | --- | --- | --- | --- | --- |

5. Entregable adicional:

Crea un dashboard sencillo en Power BI que muestre:

* Top 5 empleados con mayor salario líquido.
* Total de horas extras pagadas por tipo.
* Porcentaje de empleados con “Alerta financiera”.

**Qué debes entregar:**

* Código en Python.
* Script .sql usado para extracción (Puedes usar la LZ o cualquier motor de tu preferencia).
* Excel final con el resultado.
* Archivo del dashboard.