Prueba Técnica DBA

En **Fundación delamujer** estamos comprometidos con mejorar la calidad de vida de las mujeres colombianas y la de su familia. Como parte de nuestro proceso de selección, le invitamos a realizar la siguiente prueba técnica para optar por el cargo de *DBA*, donde se evaluarán habilidades clave necesarias para el desarrollo del rol.

Instrucciones Generales

- Esta prueba debe ser enviada a más tardar el **12 de diciembre de 2025 a las 12:30 pm horas**.
- Conteste con sus propias palabras las preguntas abiertas, no copie definiciones textuales.
- Las soluciones deben ser agnósticas a bases de datos e infraestructura de despliegue (on-premise, cloud).
- Una **vez finalizada la prueba**, envíe un pdf con la solución de la misma. Si realiza el punto opcional, comparta el enlace al repositorio de código.
- Se valorará el tiempo de entrega, la originalidad, la claridad y profundidad de las respuestas, la justificación técnica mediante experiencias reales, la aplicación de buenas prácticas, entre otras. Mencione los programas, herramientas o scripts que ha utilizado en su experiencia para desarrollar los ejercicios.

Prueba

1. **Diseñe una solución de alta disponibilidad para una base de datos crítica que actualmente corre en on-premise, pero que debe tener un entorno de contingencia en nube.**

   Implementaría Always On Availability Groups de SQL Server con la siguiente configuración:

   Arquitectura
   - **Nodo 1:** primario, on-premise
   - **Nodo 2:** secundario síncrono, on-premise (failover automático)
   - **Nodo 3:** secundario asíncrono, Azure VM (failover manual)

   Con esta configuración tenemos alta disponibilidad de manera local y un entorno de contingencia en la nube. El nodo secundario síncrono permitiría una recuperación inmediata y failover automático ante un fallo del nodo primario, garantizando cero pérdida de datos. El nodo asíncrono en Azure entraría como una alternativa de recuperación ante desastre en caso de caída total del datacenter.

-RTO (Recovery Time Objective): tiempo máximo de recuperación de hasta 30 minutos para fallos locales y 2 horas para recuperación desde la nube.
-RPO (Recovery Point Objective): pérdida de datos aceptable. Con la réplica síncrona local el RPO es 0. En el caso de la réplica en Azure dependería del ancho de banda entre los sitios, el cual podría oscilar entre 5 y 15 minutos.

La conexión entre on-premise y Azure se realizaría por VPN o enlace dedicado. Además, los backups generados on-premise se copiarían automáticamente al storage en la nube para tener un plan alterno de restauración en caso de ser necesario.

2. **¿Cómo gestionaría los backups en un sistema on-premise y cómo podría asegurar que puedan ser usados en un escenario de recuperación en la nube? ¿Qué validaciones haría para garantizar la integridad del backup restaurado?**

Para un entorno on-premise entraría a evaluar con el equipo de desarrolladores y dueños del producto la criticidad y transaccionalidad de las bases de datos para definir la periodicidad. Una buena práctica en términos generales sería realizar al menos un backup full semanal, uno diferencial diario y el de logs transaccionales cada hora, ajustando los horarios para no afectar las horas pico.

Si la empresa no cuenta con un software especializado para backups y restauraciones (como NetBackup, Networker, entre otros) optaría por automatizar todo con SQL Agent apoyándome en los scripts de mantenimiento de Ola Hallengren, por la facilidad de configuración y robustez.

En cuanto al almacenamiento de los backups deben realizarse hacia una ruta de red (nunca en el servidor local), además de subirlos a la nube y copias en cinta para una mayor seguridad.

Para asegurarme de poder restaurar estos backups en la nube establecería un cronograma periódico de pruebas de restauración en una VM de Azure con SQL Server. Así valido compatibilidad, tiempos y lectura correcta del archivo.

Para validaciones de integridad configuraría que cada backup incluya CHECKSUM y al finalizar se ejecute un RESTORE VERIFYONLY para verificar que el archivo esté en buen estado. También implementaría una tarea automática que realice DBCC CHECKDB diario a las bases de datos con informes vía correo si llegara a detectar alguna problemas de integridad.

3. **¿Cómo diagnosticaría problemas de rendimiento en una consulta que antes era rápida y ahora es lenta? ¿Qué pasos seguiría y qué herramientas usaría?**

   Lo primero que haría sería verificar si la consulta está siendo bloqueada, esto lo realizaría mediante herramientas como sp_who2, sp_whoisactive, o sp_blitzFirst para ver sesiones en espera, locks o transacciones de larga duración.

   Luego revisaría el consumo de recursos del servidor (CPU, memoria y disco) mediante el administrador de tareas o con DMVs.

   Después me enfocaría en la consulta específica. Obtendría el plan de ejecución actual mediante el SSMS y lo compararía con uno anterior si está disponible en Query Store y buscaría identificar cual instrucción está realizando más consumo de recursos. Muchas veces el problema es un cambio de plan debido a estadísticas desactualizadas, por lo que verificaría cuándo se actualizaron por última vez.

   También revisaría la fragmentación de los índices involucrados y si el tamaño de la tabla ha crecido significativamente, no es lo mismo consultar una tabla de 500 MB que una de 500 GB.

   Finalmente hablaría con los desarrolladores para validar si ha habido cambios recientes en la estructura de la tabla y por ende se requiera añadir un nuevo campo al índice.

4. **¿Cómo protegería los datos sensibles en una base de datos? Teniendo presente campos, gestión de claves, control de accesos, backups y conexiones seguras.**

   Para proteger datos sensibles primero identificaría cuales columnas hay que proteger (correos, cuentas bancarias, celulares, etc.). Con eso definiría el nivel de protección necesaria para cada una.
   A nivel de cifrado aplicaría dos capas:
   -TDE (transparent data encryption) para cifrar archivos y backups en disco.
   -Always Encrypted para columnas muy sensibles, así solo la aplicación autorizada podría leerlas.
   La gestión de claves las gestionaría fuera de la base de datos mediante Azure Key Vault.

   En cuanto al control de accesos usaría roles personalizados y aplicando privilegios mínimos específicos para los que requiere cada usuario según sus funciones.

   Los backups siempre cifrados y almacenados en rutas seguras.
   Por último, las conexiones entre aplicación y bd usando cifrado SSL/TLS, y para conexiones entre on-premise y nube asignaría una VPN o un enlace dedicado seguro.

5. **¿Cómo implementaría una estrategia de monitoreo, backup y mantenimiento para tablas críticas en entornos on-premise y cloud, en bases de datos SQL y NoSQL?**

   Para entornos SQL Server on-premise usaría SQL Agent junto con los scripts de mantenimiento de Ola Hallengren, ya que estos me permiten automatizar mantenimiento de índices, actualización de estadísticas, backups full/diferenciales/logs, configurar periodicidad de una manera sencilla y guardar resultados en tablas de historial. Sobre esos registros históricos podría generar reportes automáticos mediante un script de powershell para consolidar el estado de todas las instancias y enviar alertas por correo.

   Para SQL en la nube aprovecharía las tareas nativas de la plataforma: alertas vía Azure Monitor, métricas de rendimiento, backups automáticos y automatic tuning cuando aplique.

   En NoSql on-premise emplearía las herramientas propias de cada motor, como mongodump/mongorestore programadas mediante crons, ademas de colectores de métricas para revisar uso de cpu, locks y estado de los shards.

   En NoSql en la nube llevaría el monitoreo basado en las métricas de la plataforma, RU/s latencia y disponibilidad, configurando alertas automáticas según lo que permita el servicio.

6. **Describa una situación en base de datos en la que un índice está presente pero no es utilizado. ¿Cómo lo detectaría?**
   Esto puede ocurrir cuando la consulta usa una función sobre la columna indexada. Por ejemplo, si existe un índice sobre APELLIDO pero la consulta aplica LOWER(APELLIDO) en el where, el motor no puede usar el índice y termina haciendo un scan.

   Otra causa puede ser cuando el motor basándose en unas estadísticas desactualizadas decide que no es necesario usar el indice.

   Para detectarlo usaría la vista sys.dm_db_index_usage_stats para identificar índices que tengan varios updates pero cero seeks/scans. También utilizaría sp_blitzIndex, el cual identifica índices no utilizados.

7. **¿Cómo detectaría y resolvería deadlocks en una base de datos?**

   Para detectar deadlocks usaría principalmente Extended Events del propio sql server, creando una sesión que capture los eventos de deadlock. También podría iniciar una traza en el sql server profiler o usar scripts como sp_BlitzLock. Con esto podría identificar que sesiones estaban involucradas y cuales objetos estaban causando el deadlock.

Una vez identificado, analizaría el orden de los bloqueos y las consultas que están en conflicto. En base a lo que encuentre, plantearía soluciones como ajustar el orden en el que las transacciones acceden a las tablas. Revisaría el nivel de aislamiento y también podría reducir el tiempo de las transacciones creando o ajustando índices para que sean más eficientes.

8. **Imagine una empresa que necesita registrar información sobre su personal, las áreas internas y las ausencias de sus empleados. A continuación, se describen las entidades principales que deben representarse en la base de datos:**

   Empleados: Cada empleado tiene un nombre, un salario, una fecha en la que ingresó a trabajar, si está activo o inactivo y el departamento al que pertenece.
   Departamentos: La empresa está organizada por departamentos. Cada departamento tiene un nombre y un identificador.
   Ausencias: Se debe llevar un registro de las ausencias de cada empleado. Cada ausencia tiene una fecha, un tipo (por ejemplo: vacaciones, enfermedad, etc.) y está vinculada a un empleado. Un empleado puede tener muchas ausencias, y algunas personas pueden no tener ninguna.

```
--Creación de la base de datos
CREATE DATABASE [pruebaDBA]
 ON  PRIMARY
( NAME = N'pruebaDBA', FILENAME = N'Y:\SQL Server\Data\pruebaDBA.mdf' , SIZE = 2048KB
, FILEGROWTH = 256KB )
 LOG ON
( NAME = N'pruebaDBA_log', FILENAME = N'Y:\SQL Server\Logs\pruebaDBA_log.ldf' , SIZE =
2048KB , FILEGROWTH = 256KB  )
GO
ALTER DATABASE [pruebaDBA] SET COMPATIBILITY_LEVEL = 150
GO

/*Creación de las tablas */

USE [pruebaDBA]
GO

CREATE TABLE departamentos (
	departamento_id INT IDENTITY(1,1) PRIMARY KEY,
	nombre VARCHAR(100) NOT NULL
	);

CREATE TABLE [dbo].[empleados](
	[empleado_id] [int] IDENTITY(1,1) PRIMARY KEY,
	[primer_nombre] [varchar](50) NOT NULL,
	[segundo_nombre] [varchar](50)  NULL,
	[primer_apellido] [varchar](50) NOT NULL,
	[segundo_apellido] [varchar](50)  NULL,
	[genero] [varchar] (1) NOT NULL,
	[salario] [decimal](10, 2) NOT NULL,
	[fecha_ingreso] [date] NOT NULL,
	[activo] [bit] NOT NULL,
	[departamento_id] [int] NOT NULL,
	FOREIGN KEY (departamento_id) REFERENCES departamentos(departamento_id)
 );
```

```
CREATE TABLE [dbo].[ausencias](
      [ausencia_id] [int] IDENTITY(1,1) PRIMARY KEY,
      [empleado_id] [int] NOT NULL,
      [fecha] [date] NOT NULL,
      [tipo] [varchar](50) NULL,
      FOREIGN KEY (empleado_id) REFERENCES empleados (empleado_id)
)
```

Consultas a realizar

Una vez que tenga creada la base de datos y cargados algunos datos de prueba, deberá escribir las consultas necesarias para responder las siguientes preguntas:

- Empleados con más de una ausencia en octubre de 2024. Mostrar: nombre del empleado, cantidad de días de ausencia y el tipo de ausencia más común.

```
/*
   Empleados con más de una ausencia en octubre 2024

*/

SELECT
    -- Armo el nombre completo
    CONCAT_WS(' ',
        e.primer_nombre,
        e.segundo_nombre,
        e.primer_apellido,
        e.segundo_apellido
    ) AS nombre_empleado,

    -- Total de días que faltó
    COUNT(*) AS cantidad_dias_ausencia,

    -- Tipo de ausencia más repetido para ese empleado
    (
        SELECT TOP 1 a2.tipo
        FROM ausencias a2
        WHERE a2.empleado_id = e.empleado_id
              --aplico las fechas que estamos buscando (mes de octubre de 2024)
          AND a2.fecha >= '2024-10-01'
          AND a2.fecha <  '2024-11-01'
        GROUP BY a2.tipo
        ORDER BY COUNT(*) DESC
    ) AS tipo_ausencia_mas_comun

FROM empleados e
JOIN ausencias a
  ON a.empleado_id = e.empleado_id
  AND a.fecha >= '2024-10-01'   -- solo octubre
  AND a.fecha <  '2024-11-01'

-- Toca agrupar por empleado y nombre
GROUP BY
    e.empleado_id,
    CONCAT_WS(' ',
        e.primer_nombre,
        e.segundo_nombre,
        e.primer_apellido,
        e.segundo_apellido
    )

HAVING COUNT(*) > 1 -- solo los que faltaron más de una vez
ORDER BY cantidad_dias_ausencia DESC, nombre_empleado;
```

	nombre_empleado	cantidad_dias_ausencia	tipo_ausencia_mas_comun
1	Lizeth Patricia Acosta	4	Personal
2	Carmen Acevedo Rodriguez	3	Enfermedad
3	Erika Marcela Aguas	2	Vacaciones
4	Jhon Elles Alforja Florez	2	Vacaciones
5	Juan Anaya Mesa	2	Personal
6	Julian Aleman Ortega	2	Personal

- Comparación salarial por departamento. Para cada empleado, mostrar su nombre, el salario promedio de su departamento y cuánto está por encima o por debajo de ese promedio.

```sql
WITH activos AS (
    SELECT *
    FROM empleados
    WHERE activo = 1  -- trabajo solo con empleados activos
)

SELECT
    -- Nombre completo armado de forma segura
    CONCAT_WS(' ',
        e.primer_nombre,
        e.segundo_nombre,
        e.primer_apellido,
        e.segundo_apellido
    ) AS nombre_empleado,

    d.nombre AS departamento,  -- nombre del departamento

    e.salario AS salario_empleado,  -- salario del empleado

    -- Promedio del departamento usando función de ventana
    AVG(e.salario) OVER (PARTITION BY e.departamento_id)
        AS salario_promedio_departamento,

    -- Diferencia entre su salario y el promedio del depto
    e.salario - AVG(e.salario) OVER (PARTITION BY e.departamento_id)
        AS diferencia_vs_promedio
FROM activos e
INNER JOIN departamentos d ON d.departamento_id = e.departamento_id
ORDER BY e.departamento_id, nombre_empleado;
```

Results Messages

	nombre_empleado	departamento	salario_empleado	salario_promedio_departamento	diferencia_vs_promedio
1	Carmen Acevedo Rodriguez	Compras	2700000	2229375	470625
2	Diana Herrera Aguas Suarez	Compras	2135250	2229375	-94125
3	Liliana Milena Agamez Gomez	Compras	2135250	2229375	-94125
4	Lizeth Patricia Acosta	Compras	2135250	2229375	-94125
5	Sergio Alberto Aguas Meza	Compras	2135250	2229375	-94125
6	Yerly Johanna Acosta Rincon	Compras	2135250	2229375	-94125
7	Agueda Aguilar Sanchez	Publicidad	1779375	1863068	-83693
8	Bibiana Luz Amaya Giron	Publicidad	2700000	1863068	836932
9	Carlos Alcoser Rico	Publicidad	1779375	1863068	-83693
10	Fernando Jairo Alfaro Sandoval	Publicidad	1779375	1863068	-83693
11	Henry Eliecer Ahumada Hernandez	Publicidad	1779375	1863068	-83693
12	Jaime Enrique Aguilar Quintero	Publicidad	1779375	1863068	-83693
13	Jose Fernando Agudelo Alquichire	Publicidad	1779375	1863068	-83693

- Empleados sin ausencias en octubre de 2024. Mostrar: nombre del empleado.

```sql
/* ●   Empleados sin ausencias en octubre de 2024. Mostrar: nombre del empleado. */

select
-- Armo el nombre completo
    CONCAT_WS(' ',
        e.primer_nombre,
        e.segundo_nombre,
        e.primer_apellido,
        e.segundo_apellido
    ) AS nombre_empleado

       from empleados e
       --verifico que el empleado se encuentre activo
       where e.activo= 1
       --comparo que la id del empleado no tenga registros en la tabla de ausencias en
octubre de 2024
       AND NOT EXISTS (
       Select *
       FROM ausencias a
       where a.empleado_id = e.empleado_id
              AND a.fecha >= '2024-10-01'
              AND a.fecha <= '2024-10-31'
              )
       order by empleado_id asc
```

	nombre_empleado
1	Jennifer Marcela Acevedo Mantilla
2	Diana Xiomara Aconcha Niño
3	Diana Herrera Aguas Suarez
4	Lorenzo Lizeth Agudelo
5	Fernando Jairo Alfaro Sandoval
6	Flor Carolina Alfaro Viana
7	Nohema Alquichire Arevalo
8	Diogenes Humberto Alvarez Serna
9	Jayson Coimbra Alvarez Cuartas
10	Jose EDUARDO Alvarez Diaz
11	Tirzo Jaime Alvear Ardila
12	Virginia Amado Galindo

Consideraciones

- Puede elegir cualquier entorno para trabajar: local, en la nube, contenedor, etc.
- Puede usar cualquier motor de base de datos que se ajuste al ejercicio.

Opcional:
El desarrollo del punto 8 será especialmente valorado si se presenta como una solución funcional y bien organizada en un repositorio de código, incluyendo los scripts, las consultas y la documentación correspondiente.