Prueba técnica para **Product Owner** de Operations&**Data**

**Objetivo:** Evaluar las habilidades y conocimientos del candidato para el puesto de Product Owner de un equipo enfocado a la integración de datos.

Test de conocimientos

1. **Data Lake**
    - Organización de los datos.
    - Gestión de datos sensibles.
    - Movimientos entre capas.

**. Data Lake**

**Organización de los datos**

- Se organiza en **capas** (Raw, Cleansed, Curated/Gold) para mejorar la gobernanza y la calidad.
- Puede seguir una estructura basada en **zonas**:
    - **Raw Zone**: Datos en su formato original.
    - **Cleansed Zone**: Datos limpios, con transformaciones mínimas.
    - **Curated Zone**: Datos listos para el consumo analítico.

**Gestión de datos sensibles**

- **Encriptación** en reposo y en tránsito.
- **Acceso basado en roles (RBAC)** en Azure Storage.
- **Anonimización o enmascaramiento** de datos personales.
- **Data Lineage** para trazabilidad.

**Movimientos entre capas**

- **ETL o ELT** con herramientas como **Azure Data Factory o Databricks**.
- Uso de **Delta Lake** para versionado y control de calidad.

2. **Data warehouse**
    - Definición de data warehouse y sus componentes principales.
    - Diferencias entre data warehouse, data mart, data lake y datalakehouse.
    - Modelos de datos dimensionales y relacionales.
    - ETL (Extracción, Transformación y Carga) y sus fases.

**Definición y componentes**

- Un **Data Warehouse (DWH)** es una base de datos optimizada para análisis. Componentes clave:
    - **Staging Area**: Capa temporal para carga de datos.
    - **Data Storage**: Modelo de datos estructurado (estrella, copo de nieve, etc.).
    - **Processing Engine**: Procesamiento de consultas optimizadas.
    - **Presentation Layer**: Acceso para reportes y dashboards.

**Diferencias clave**

| Característica | Data Lake | Data Warehouse | Data Mart | Data Lakehouse |
|---|---|---|---|---|
| Formato de datos | No estructurado / semi | Estructurado | Estructurado | Mixto |
| Costo | Bajo (almacenamiento) | Alto (procesamiento) | Moderado | Moderado |
| Uso principal | Big Data, ML | Reporting, BI | Reporting departamental | Híbrido |
| Tecnología común | ADLS, Databricks | Synapse Analytics | Power BI | Delta Lake |

**Modelos de datos**

- **Relacional**: Tablas normalizadas (3NF).
- **Dimensional**: Modelo estrella/copo de nieve para optimización analítica.

**ETL (Extract, Transform, Load)**

- **Extracción**: Desde APIs, BBDD, Data Lakes.
- **Transformación**: Limpieza, agregaciones, modelado.
- **Carga**: DWH o Data Marts (Full o Incremental).

Herramientas en **Azure**: **Data Factory, Synapse Pipelines, Databricks**.

3. **Modelado de datos**
    - Entidades, atributos y relaciones en el modelado de datos.
    - Diagramas de entidad-relación (E-R) y notación Crow's Foot.
    - Normalización de datos y sus diferentes niveles.
    - Diseño de dimensiones y hechos en un data warehouse.
    - Componentes básicos del modelado Data Vault.

**Entidades, atributos y relaciones**

- **Entidades**: Objetos (Ej. Cliente, Producto).
- **Atributos**: Propiedades (Ej. Nombre, Precio).
- **Relaciones**: Vínculos entre entidades.

**Diagramas E-R y Notación Crow's Foot**

- Modelo visual para representar relaciones.
- Notación Crow's Foot indica cardinalidad (*1:1, 1:M, M:M*).

**Normalización**

- **1NF**: Eliminar duplicados, columnas atómicas.
- **2NF**: No dependencias parciales.
- **3NF**: No dependencias transitivas.

**Dimensiones y Hechos**

- **Hechos**: Eventos medibles (*ventas, transacciones*).
- **Dimensiones**: Contexto para análisis (*tiempo, cliente, producto*).

**Modelado Data Vault**

- **Hubs** (entidades centrales).

- **Links** (relaciones entre hubs).
- **Satélites** (atributos de hubs y links).

4. **SQL - Transformaciones** (sintaxis)
    - Consultas básicas de SQL (SELECT, FROM, WHERE, JOIN, ORDER BY, GROUP BY).
    - Agregación de datos (SUM, COUNT, AVG, MIN, MAX).
    - Subconsultas y correlaciones.
    - Funciones de ventana.
    - Manipulación de cadenas y fechas en SQL.

    4_SQL - Transformaciones (sintaxis).txt

5. **Ecosistema Azure**
    - ¿Qué componentes del Ecosistema de Azure usarías para cada uno de los puntos de los bloques anteriores y por qué motivo?

    ¿Qué componentes usarías en cada bloque?

    Data Lake → Azure Data Lake Storage (ADLS) para almacenamiento.

    Data Warehouse → Azure Synapse Analytics para análisis estructurado.

    ETL → Azure Data Factory o Databricks para transformación de datos.

    Modelado → SQL Database, Synapse o Power BI para diseño de modelos.

    SQL → Azure SQL Database o Synapse SQL Pools para ejecutar consultas.

## Prueba técnica

**Caso de uso:** Empresa de venta de seguros

**Objetivo:** Evaluar la capacidad del candidato para aplicar sus conocimientos y habilidades a un caso de negocio real en el ámbito de la integración de datos para el sector seguros.

**Descripción:**

La empresa de seguros "Seguros XYZ" está buscando modernizar su infraestructura de datos para mejorar la toma de decisiones y ofrecer un mejor servicio al cliente. Se requiere un equipo de integración de datos para desarrollar un **data warehouse** que centralice la información de diferentes fuentes, incluyendo:

- **Sistema de gestión de clientes:** Datos de los clientes, como nombre, dirección, teléfono, pólizas contratadas, etc.
- **Sistema de siniestros:** Datos de los siniestros ocurridos, como fecha, causa, importe, etc.
- **Datos externos:** Datos demográficos, económicos y meteorológicos.

El data warehouse debe permitir a la empresa realizar análisis de datos para:

- **Identificar clientes con mayor riesgo de siniestralidad.**
- **Detectar patrones de fraude.**
- **Desarrollar nuevos productos y servicios de seguros.**
- **Mejorar la atención al cliente.**

## Casos prácticos:

**1. Ejercicio práctico de modelado de datos (*registrar el tiempo invertido y el porcentaje de uso de GPT*)**

- A partir de la descripción del caso de uso, el candidato deberá diseñar un modelo de datos dimensional para el data warehouse de Seguros XYZ.
- El modelo de datos debe incluir las entidades, atributos, relaciones, dimensiones y hechos necesarios para dar soporte a los análisis de datos descritos anteriormente.
- Se valorará la capacidad del candidato para:
    - Identificar las entidades relevantes para el negocio.
    - Definir correctamente los atributos de cada entidad.
    - Establecer relaciones entre las entidades de forma clara y concisa.
    - Diseñar dimensiones y hechos adecuados para los análisis de datos.
    - Utilizar un diagrama de entidad-relación (E-R) para representar el modelo de datos.
    - Modelar en capas de Raw Data Vault (RDV) y Business Data Vault (BDV), así como la construcción del Information Mart.

Data Warehouse Model for Seguros XYZ

1. Raw Data Vault (RDV)

Hubs:

- Hub_Clients (Client_ID, Client_Name, Address, Phone, Birth_Date, Created_At)

- Hub_Policies (Policy_ID, Client_ID, Policy_Type, Start_Date, End_Date, Created_At)
- Hub_Claims (Claim_ID, Policy_ID, Claim_Date, Cause, Created_At)
- Hub_External_Data (External_Data_ID, Data_Type, Data_Source, Created_At)

Links:

- Link_Client_Policy (Client_ID, Policy_ID, Created_At)
- Link_Policy_Claim (Policy_ID, Claim_ID, Created_At)

Satellites:

- Sat_Clients (Client_ID, Risk_Score, Income_Level, Last_Updated_At)
- Sat_Policies (Policy_ID, Premium_Amount, Coverage_Details, Last_Updated_At)
- Sat_Claims (Claim_ID, Claim_Amount, Claim_Status, Last_Updated_At)
- Sat_External_Data (External_Data_ID, Economic_Index, Weather_Conditions, Last_Updated_At)

2. Business Data Vault (BDV)

Business Hubs:

- Hub_Risk_Analysis (Client_ID, Risk_Category, Created_At)
- Hub_Fraud_Detection (Claim_ID, Fraud_Score, Created_At)

Business Links:

- Link_Client_Risk (Client_ID, Risk_Category, Updated_At)
- Link_Claim_Fraud (Claim_ID, Fraud_Score, Updated_At)

Business Satellites:

- Sat_Risk_Analysis (Client_ID, Risk_Factors, Adjusted_Score, Last_Updated_At)
- Sat_Fraud_Detection (Claim_ID, Fraud_Patterns, Investigated_By, Last_Updated_At)

3. Information Mart (Dimensional Model)

Fact Tables:

- Fact_Claims (Claim_ID, Policy_ID, Client_ID, Claim_Date, Claim_Amount, Claim_Status, Fraud_Score, Risk_Score)
- Fact_Policies (Policy_ID, Client_ID, Premium_Amount, Start_Date, End_Date, Policy_Type)
- Fact_Customer_Risk (Client_ID, Risk_Score, Number_of_Claims, Premium_Amount, Economic_Index, Weather_Conditions)

Dimension Tables:

- Dim_Clients (Client_ID, Client_Name, Address, Phone, Birth_Date, Risk_Category)
- Dim_Policies (Policy_ID, Policy_Type, Premium_Amount, Coverage_Details)
- Dim_Claims (Claim_ID, Claim_Date, Claim_Amount, Cause, Claim_Status, Fraud_Score)
- Dim_External_Factors (External_Data_ID, Economic_Index, Weather_Conditions)

**2. Ejercicio práctico de ETL con Python/SQL (*registrar el tiempo invertido*)**

- El candidato deberá desarrollar un script en Python/PySpark/T-SQL para extraer datos del sistema de gestión de clientes de Seguros XYZ, transformarlos y cargarlos en el data warehouse.
- El script debe incluir la conexión a la base de datos del sistema de gestión de clientes, la lectura de los datos, la limpieza y transformación de los datos, y la carga de los datos en el data warehouse.
- Se valorará la capacidad del candidato para:
    - Construir un plan de tareas e hitos asociados a este desarrollo y evoluciones futuras.
    - Conectarse a una base de datos mediante Python / PySpark.
    - Leer y manipular datos de forma eficiente.
    - Aplicar técnicas de limpieza y transformación de datos.
    - Definición del ciclo de vida del dato y las capas de persistencia estandarizadas en el movimiento de los datos.
        - **Ciclo de Vida del Dato**
        - El ciclo de vida del dato describe las etapas por las que pasan los datos desde su creación hasta su eliminación. Estas etapas aseguran que los datos se gestionen de manera eficiente y segura a lo largo de su existencia. Las principales fases del ciclo de vida del dato son:
            - **Generación o Captura**: Los datos se crean o se capturan desde diversas fuentes, como sensores, transacciones, encuestas, etc.
            - **Recolección**: Los datos se identifican, etiquetan y registran para su uso posterior.
            - **Almacenamiento**: Los datos se guardan en sistemas de almacenamiento adecuados, como bases de datos, data lakes, o data warehouses.
            - **Procesamiento**: Los datos se limpian, transforman y analizan para extraer información útil.
            - **Utilización**: Los datos procesados se utilizan para tomar decisiones, generar informes, o alimentar aplicaciones.
            - **Archivado**: Los datos que ya no se utilizan activamente se archivan para su posible uso futuro.
            - **Eliminación**: Los datos se eliminan de manera segura cuando ya no son necesarios, cumpliendo con las políticas de retención de datos.
        - **Capas de Persistencia**
        - Las capas de persistencia se refieren a los diferentes niveles en los que se almacenan y gestionan los datos a lo largo de su ciclo de vida. Estas capas aseguran que los datos se manejen de manera eficiente y segura en cada etapa. Las capas de persistencia estandarizadas incluyen:
            - **Capa de Ingesta**: Aquí se capturan y almacenan los datos en bruto desde diversas fuentes. Esta capa se encarga de la ingesta inicial de datos.
            - **Capa de Procesamiento**: En esta capa, los datos se limpian, transforman y enriquecen. Se realizan operaciones de ETL (Extract, Transform, Load) para preparar los datos para su análisis.

- **Capa de Almacenamiento**: Los datos procesados se almacenan en sistemas de almacenamiento adecuados, como data warehouses o data lakes, para su análisis y consulta.
- **Capa de Análisis**: En esta capa, los datos se analizan y se utilizan para generar informes, dashboards y modelos analíticos.
- **Capa de Presentación**: Los resultados del análisis se presentan a los usuarios finales a través de aplicaciones, informes y dashboards interactivos.
- **Capa de Archivado**: Los datos que ya no se utilizan activamente se archivan para su posible uso futuro, asegurando su disponibilidad y cumplimiento con las políticas de retención de datos.

Estas capas de persistencia permiten una gestión estructurada y eficiente de los datos, asegurando que se manejen de manera adecuada en cada etapa de su ciclo de vida.

- Cargar datos en un data warehouse.
- Escribir código claro, conciso y bien documentado.
- El uso estandarizado en un repositorio de GitHub y política de ramas correcta.

**Plan de Trabajo:**
**Fase 1: Diseño y planificación**

1. **Definir fuentes de datos** en *database1* (CRM) → Clients, Policies, Claims, External Factors.
2. **Identificar las transformaciones necesarias**, como limpieza de datos, formateo de fechas, normalización de nombres.
3. **Definir la arquitectura del ETL**:
   - **Extracción** desde database1 (Azure Synapse Lake Database/ archivos de datos en el Data Lake de Azure).
   - **Transformación** con PySpark (limpieza, formateo, enriquecimiento).
   - **Carga** en insuranse_xyz_DW.
4. **hitos y evolución futura**:
   - Desarrollo del script ETL inicial.
   - Automatización mediante Azure Data Factory.
   - Monitoreo y optimización de rendimiento.

Conectarse a una base de datos mediante Python / PySpark.
Leer y manipular datos de forma eficiente.
Aplicar técnicas de limpieza y transformación de datos.
Cargar datos en un data warehouse.
Escribir código claro, conciso y bien documentado.

**1. Importar las librerías necesarias**

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, to_date, current_timestamp
```

Estas líneas importan las librerías necesarias de PySpark para crear una sesión de Spark y trabajar con funciones de DataFrame.

**2. Configurar la sesión de Spark**

```
spark = SparkSession.builder \
  .appName("ETL_Insurance_XYZ") \
  .getOrCreate()
```

Aquí se configura y se crea una sesión de Spark con el nombre de la aplicación "ETL_Insurance_XYZ". Esta sesión es necesaria para ejecutar operaciones de Spark.

**3. Definir la ruta base en Data Lake**

```
base_path = "abfss://data@azdatadp900.dfs.core.windows.net/raw-data/CRM_SAMPLE/"
```

Se define la ruta base donde se encuentran los archivos de datos en el Data Lake de Azure.

**4. Leer datos desde los archivos en Data Lake**

```
clients_df = spark.read.option("header", "true").csv(base_path + "clients.csv")
policies_df = spark.read.option("header", "true").csv(base_path + "policies.csv")
claims_df = spark.read.option("header", "true").csv(base_path + "claims.csv")
external_factors_df = spark.read.option("header", "true").csv(base_path + "external_factors.csv")
```

Estas líneas leen los archivos CSV desde el Data Lake y los cargan en DataFrames de Spark. La opción header se establece en true para indicar que los archivos CSV tienen una fila de encabezado.

**5. Función de limpieza de datos**

```
def clean_dataframe(df):
    return df.dropDuplicates().na.fill("Unknown")
```

Se define una función clean_dataframe que elimina duplicados y rellena valores nulos con "Unknown".

**6. Limpiar los DataFrames**

```
clients_df = clean_dataframe(clients_df)
policies_df = clean_dataframe(policies_df)
claims_df = clean_dataframe(claims_df)
external_factors_df = clean_dataframe(external_factors_df)
```

Se aplica la función de limpieza a cada uno de los DataFrames.

**7. Convertir columnas de fecha**

```
claims_df = claims_df.withColumn("Claim_Date", to_date(col("Claim_Date"), "yyyy-MM-dd"))
policies_df = policies_df.withColumn("Start_Date", to_date(col("Start_Date"), "yyyy-MM-dd"))
policies_df = policies_df.withColumn("End_Date", to_date(col("End_Date"), "yyyy-MM-dd"))
```

Estas líneas convierten las columnas de fecha en los DataFrames claims_df y policies_df al formato de fecha yyyy-MM-dd.

**8. Agregar columna de timestamp de carga**

```
for df in [clients_df, policies_df, claims_df, external_factors_df]:
    df = df.withColumn("Load_Timestamp", current_timestamp())
```

Se agrega una columna Load_Timestamp a cada DataFrame, que contiene la marca de tiempo actual cuando se carga el DataFrame.

**9. Escribir los datos en el Data Warehouse bajo el esquema dbo**

```
clients_df.write.mode("overwrite").format("delta").saveAsTable("default.Hub_clients")
policies_df.write.mode("overwrite").format("delta").saveAsTable("default.Hub_policies")
claims_df.write.mode("overwrite").format("delta").saveAsTable("default.Hub_claims")
external_factors_df.write.mode("overwrite").format("delta").saveAsTable("default.Hub_external_factors")
```

Estas líneas escriben los DataFrames en el Data Warehouse utilizando el formato Delta y el modo overwrite, lo que significa que cualquier dato existente en las tablas será sobrescrito. Las tablas se guardan en el esquema default con los nombres especificados.

**10. Mensaje de finalización**

```
print("ETL completado exitosamente")
```