Ejercicio de Data Engineer

Contexto

Para desenvolverse de forma ágil en el puesto, poder manejarse con Python y, en general, ser capaz de manipular datos con SQL resultan habilidades esenciales.
Se proponen una serie de ejercicios que nos permitan entender un poco más de qué manera se encararía la solución de algunos requerimientos que pueden surgir en el día a día.

Restricciones

- Hacer lo más simple que pueda funcionar.
- Escribir la mejor solución que se pueda.
- No hacer más de lo que pide la funcionalidad.

Ejercicio 1 - Manejo de Datos (Usando sintaxis Redshift)

Esquema de Tablas:

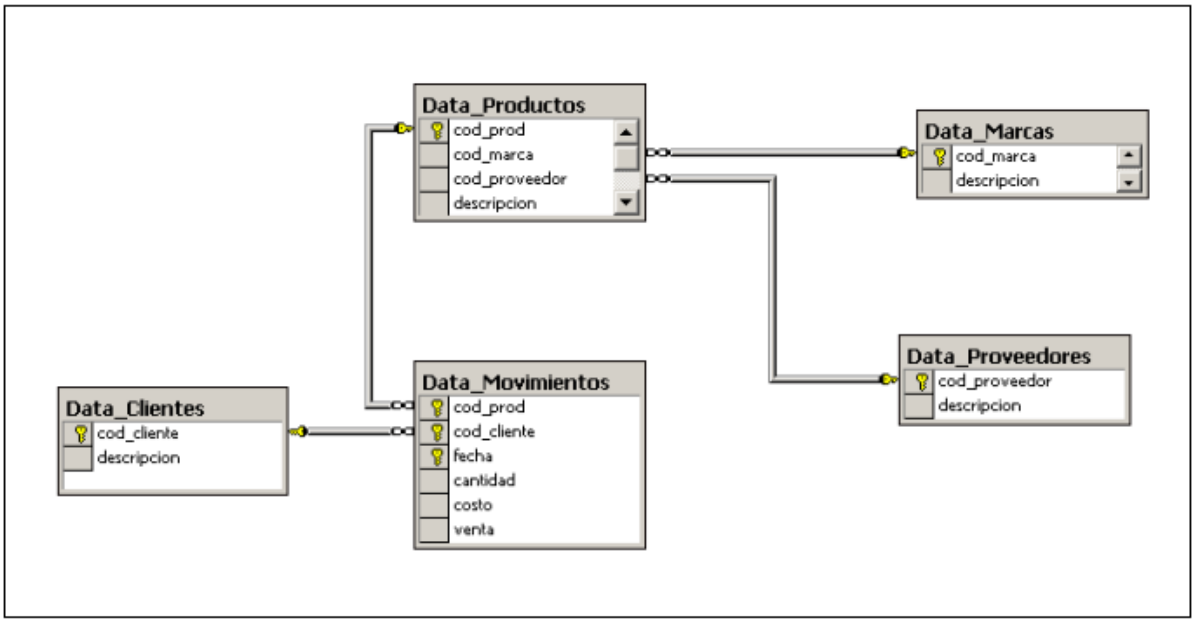
Data_Productos
Cod_Prod (Numérico)
Cod_Marca (Numérico)
Cod_Proveedor (Numérico)
Descripcion (Texto)

Data_Marcas
Cod_Marca (Numérico)
Descripcion (Texto)

Data_Proveeedores
Cod_Proveedor (Numérico)
Descripcion (Texto)

Data_Clientes
Cod_Cliente (Numérico)
Descripcion (Texto)

Data_Movimientos
Cod_Prod (Numérico)
Cod_Cliente (Numérico)
Fecha (DateTime)
Cantidad (Numérico)
Costo (Numérico)
Venta (Numérico)



Nota:
En la tabla Data_Movimientos, el campo Venta es el precio total de la venta, por lo que costo/cantidad da el costo unitario de cada producto en cada operación, y venta/cantidad da el precio de venta unitario en cada operación.

a- Generar una tabla, con el listado de todos los Movimientos, con el siguiente contenido :

. Fecha
. Descripción de Cliente
. Descripción de Proveedor
. Descripción de Producto
. Descripcion de Marca
. Cantidad
. Costo
. Venta
. Ganancia Neta

b- Mostrar un listado de todas las Marcas que no tuvieron Ventas.

c- En base a la tabla generada en a, consultar, ordenando por fecha y descripción del cliente:

. Fecha
. Descripción de Cliente
. Ganancia Neta Acumulada en las últimas 7 operaciones

La idea del punto c es, dado un cliente y una fecha de operación, mostrar la sumatoria de las ganancias derivadas de las últimas siete operaciones que haya realizado.

Ejercicio 2 - Algo de Python

Se deberá escribir un script que transforme el archivo datos_data_engineer.tsv en un archivo CSV que pueda ser insertado en una base de datos, y/o interpretado por cualquier parser estándar de archivos delimitados, de la manera más simple posible.

El archivo resultante debe tener las siguientes características:

- Cada row contiene la misma cantidad de campos
- Los campos se separan con un pipe |
- Se deben poder leer correctamente los caracteres especiales que estén presentes en los campos actuales del archivo.
- El encoding del archivo final debe ser UTF-8 (datos_data_engineer.tsv es un archivo UTF-16LE)

Consideraciones

- Otorgarle estructura al código, para que el mismo quede ordenado y bien organizado.
- El código o módulo resultante debe poder ser reutilizable.

Ejercicio 3 - Preguntas en general (Soluciones de pocas líneas)

a- ¿Qué formas de hacer scheduling de una tarea en linux conoce?

b- ¿Cómo y con qué comandos guardaría la mayor cantidad de detalle sobre las salidas de un script python que desea ejecutar de forma diaria a las 6AM?

c- ¿Qué comando o serie de comandos utilizaría para subir todos los contenidos de un directorio a un bucket de S3?

d- Si una instancia de Redshift utilizada para reporting se está quedando sin espacio y se impone la necesidad de sacar algunos datos antiguos de la base, pero a pesar de que los datos de más de seis meses de antigüedad no se utilicen para reporting, se los requiere para entrenar y validar modelos predictivos, además de hacer algunos análisis ad-hoc en SQL **a un precio razonable considerando tanto infraestructura como costos de consultas** ¿Que tipo de solución propondría para poder consultar los datos usando servicios cloud en AWS? ***Intentar ser lo más descriptivo posible.***

Formatos de entrega válidos

- Solución en repositorio privado (bitbucket, gitlab, otros), con algunos commits descriptivos.