1. **Construcción de query en SQL**

**Instrucciones:** Con el diagrama de base de datos compartido, crea un query para obtener la información de ventas del cliente, incluyendo los siguientes campos:

* # de ticket
* Fecha
* Monto (DCANT) con y sin IVA
* Vendedor
* Cliente
* Productos comprados (SKU)
* Descripción
* Cantidad (unidades)
* Almacén que vendió
* Precio de lista
* Talla
* Color
* Temporada



**Entregable:** Código escrito con la consulta solicitada. Puedes asumir que las tablas ya están creadas en la base de datos.

SELECT

t.ticket\_id AS "# de ticket",

t.fecha AS "Fecha",

t.monto\_con\_iva AS "Monto con IVA",

t.monto\_sin\_iva AS "Monto sin IVA",

v.nombre AS "Vendedor",

c.nombre AS "Cliente",

p.sku AS "Productos comprados (SKU)",

p.descripcion AS "Descripción",

tp.cantidad AS "Cantidad (unidades)",

a.nombre AS "Almacén que vendió",

p.precio\_lista AS "Precio de lista",

p.talla AS "Talla",

p.color AS "Color",

p.temporada AS "Temporada"

FROM

Tickets t

JOIN

Clientes c ON t.cliente\_id = c.cliente\_id

JOIN

Vendedores v ON t.vendedor\_id = v.vendedor\_id

JOIN

Ticket\_Productos tp ON t.ticket\_id = tp.ticket\_id

JOIN

Productos p ON tp.producto\_id = p.producto\_id

JOIN

Almacenes a ON t.almacen\_id = a.almacen\_id;

1. **Desarrollo de visualizaciones en Business Intelligence**

**Instrucciones:** Con el archivo “Fuente para dashboard”, desarrolla un dashboard que permita responder la mayor cantidad de preguntas de la siguiente lista, para dar una recomendación al cliente acerca de su operación.

**Preguntas:**

1. ¿Qué materiales me voy a quedar sin inventario y en qué fecha me quedo sin inventario?
2. ¿Cuáles productos son los más importantes en venta?
3. ¿Cuáles productos tengo riesgo de quedarme con mucho inventario al final del año?
4. ¿Hay mucha variación entre la venta de años pasados y el forecast?
5. ¿Cuál sería la cobertura del inventario contra la versión 1 del forecast y con la versión 2 del forecast?
6. Después de haber presentado esta primera versión del dashboard, ¿qué pasos recomendarías seguir para liberarlo para el uso en el día a día por parte del cliente?
7. ¿Cómo esperarías que fuera la interacción entre IBSO y el cliente por los siguientes 12 meses?

**Entregable:** Dashboard (En Tableau / Power BI / Qlik) con las tablas, gráficas y/o visualizaciones que justifiquen los resultados y recomendaciones propu´x estas. El dashboard debería seguir una estructura lógica y deberá permitir al usuario entenderlo sin mucha capacitación.

**Referencias para estructura del dashboard:**

<https://medium.com/@mokkup/8-essential-dashboard-design-principles-for-effective-data-visualization-40653c5fd135>

Tableau: <https://data.ucop.edu/support-training/tableau-files/building_effective_dashboards.pdf>

1. **Ejercicios – Python**

**Instrucciones:** Con las preguntas mostradas, trata de desarrollar los códigos para responder el mayor número de ejercicios usando Python.

**Entregable:** Códigos en Python mostrando los resultados de los ejercicios.

**Ejercicio 1:**

1. Crea un diccionario llamado datos\_climaticos donde cada clave es el nombre de una ciudad y el valor es una lista de temperaturas (en grados Celsius) registradas a lo largo de una semana.
2. Para cada ciudad, calcula la temperatura promedio, la temperatura máxima y la mínima de la semana.
3. Determina cuál fue la ciudad con la temperatura promedio más alta y la más baja durante la semana.

def analizar\_datos\_climaticos(datos\_climaticos):

"""

Calcula estadísticas climáticas para cada ciudad en el diccionario.

Args:

datos\_climaticos: Un diccionario donde las claves son ciudades y los valores son listas de temperaturas.

Returns:

Un diccionario con las estadísticas de cada ciudad (promedio, máxima, mínima).

"""

resultados = {}

for ciudad, temperaturas in datos\_climaticos.items():

promedio = sum(temperaturas) / len(temperaturas)

maxima = max(temperaturas)

minima = min(temperaturas)

resultados[ciudad] = {'promedio': promedio, 'maxima': maxima, 'minima': minima}

return resultados

# Ejemplo de uso:

datos\_climaticos = {

"Ciudad de México": [22, 23, 24, 25, 26, 25, 24],

"Nueva York": [15, 18, 20, 17, 16, 19, 15],

"Tokio": [28, 30, 32, 31, 29, 28, 27]

}

resultados = analizar\_datos\_climaticos(datos\_climaticos)

# Encontrar la ciudad con la temperatura promedio más alta y más baja

ciudad\_mas\_caliente = max(resultados, key=lambda x: resultados[x]['promedio'])

ciudad\_mas\_fria = min(resultados, key=lambda x: resultados[x]['promedio'])

print("Resultados:")

for ciudad, datos in resultados.items():

print(f"{ciudad}: Promedio={datos['promedio']:.1f}°C, Máxima={datos['maxima']}°C, Mínima={datos['minima']}°C")

print(f"La ciudad con la temperatura promedio más alta fue: {ciudad\_mas\_caliente}")

print(f"La ciudad con la temperatura promedio más baja fue: {ciudad\_mas\_fria}")

**Ejercicio 2:**

Asigna a cada letra minúscula un valor, desde 1 para la 'a' hasta 26 para la 'z'.

Crea una función que pida al usuario una cadena de letras minúsculas y responde la suma de los valores de las letras en la cadena. (Ejemplo: hola = 8 + 15 + 12 + 1 = 36).

Si el usuario te da un número o una letra mayúscula, pídele que lo cambie (Input: Hola. Output: Cambia a minúscula la letra “H” en la posición 1. Input: int2. Output: Cambia el número en la posición 4 por una letra minúscula).

def valor\_cadena(cadena):

"""

Calcula el valor numérico de una cadena de letras minúsculas.

Args:

cadena: Una cadena de letras minúsculas.

Returns:

La suma de los valores numéricos de las letras.

"""

valor\_total = 0

for letra in cadena:

if letra.islower():

valor\_total += ord(letra) - 96 # Asigna valores del 1 al 26 a las letras minúsculas

else:

print(f"La letra '{letra}' en la posición {cadena.index(letra) + 1} no es minúscula. Por favor, intenta de nuevo.")

return None # Detenemos el cálculo si hay una letra no válida

return valor\_total

while True:

cadena = input("Ingrese una cadena de letras minúsculas: ")

resultado = valor\_cadena(cadena)

if resultado is not None:

print("El valor total de la cadena es:", resultado)

break

**Ejercicio 3:**

1. Extraer la información del csv **Prueba\_Promociones**

2. ⁠Generar un código donde el usuario pueda ingresar las siguientes variables

- fecha inicio (convertir a datetime)

- ⁠fecha fin (convertir a datetime)

- ⁠categoría (validar que sea string)

- ⁠uso (validar que sea string)

- ⁠sku (no permitir al usuario avanzar si no ingreso un valor string en el campo de SKU)

- ⁠% (validar que sea número decimal)

- ⁠inventario inicial (validar que sea entero)

3. Generar una nueva columna donde se coloque el # de semana correspondiente de la fecha

4. ⁠Con él % ingresado por el usuario, impactar ese porcentaje como crecimiento (columna de piezas) para todos los datos que cumplan con las siguientes condiciones:

- estén dentro del rango de fecha seleccionado. Si el usuario no coloca fecha fin entonces desde fecha inicio hasta el final, si el usuario no coloca fecha inicio entonces desde la fecha fin hasta el principio y si no coloca ninguna, a todas las fechas disponibles

- ⁠la columna **modelo** sea diferente a “real”

- ⁠la columna **uso** sea igual al valor ingresado por el usuario, en caso de que esté vacío ese campo, la columna categoría sea igual al valor ingresado por el usuario. En caso de que ambas vengan vacías aplicar a todo

5. Una vez aplicado el incremento %, generar un nuevo dataframe en donde solo venga información del SKU seleccionado y se pueda visualizar el consumo de inventario, es decir, el inventario inicial colocado por el usuario será la cantidad inicial el 01-01-2024 y de ahí se tendrá que ir consumiendo (restando) la venta proyectada (**modelo** diferente de real). Se deberá imprimir la primera fecha en la que se vuelve negativo el inventario

import pandas as pd

from datetime import datetime

def analizar\_promociones(archivo, fecha\_inicio, fecha\_fin, categoria, uso, sku, porcentaje, inventario\_inicial):

# Leer el archivo CSV

df = pd.read\_csv(archivo)

# Convertir fechas a datetime

df['fecha'] = pd.to\_datetime(df['fecha'])

fecha\_inicio = pd.to\_datetime(fecha\_inicio)

fecha\_fin = pd.to\_datetime(fecha\_fin)

# Calcular el número de semana

df['semana'] = df['fecha'].dt.isocalendar().week

# Aplicar el crecimiento

condiciones = [

(df['fecha'] >= fecha\_inicio) & (df['fecha'] <= fecha\_fin),

(df['modelo'] != 'real'),

(df['uso'] == uso) | (df['categoria'] == categoria)

]

choices = [True, False, False]

df['aplicar\_crecimiento'] = np.select(condiciones, choices, default=True)

df['ventas\_proyectadas'] = df['ventas'] \* (1 + porcentaje/100) \* df['aplicar\_crecimiento']

# Crear DataFrame para el SKU seleccionado

df\_sku = df[df['sku'] == sku].copy()

df\_sku['inventario'] = inventario\_inicial - df\_sku['ventas\_proyectadas'].cumsum()

# Encontrar la primera fecha con inventario negativo

primera\_fecha\_negativa = df\_sku[df\_sku['inventario'] < 0]['fecha'].min()

return df\_sku, primera\_fecha\_negativa

# Ejemplo de uso

archivo = "csvPrueba\_Promociones.csv"

fecha\_inicio = "2023-11-01"

fecha\_fin = "2023-12-31"

categoria = "Electronica"

uso = "Hogar"

sku = "SKU123"

porcentaje = 10

inventario\_inicial = 1000

df\_resultado, primera\_fecha\_negativa = analizar\_promociones(archivo, fecha\_inicio, fecha\_fin, categoria, uso, sku, porcentaje, inventario\_inicial)

print(df\_resultado)

print("Primera fecha con inventario negativo:", primera\_fecha\_negativa)

**Ejercicio 4: Recomendador de libros**

Fuente: <https://developer.nytimes.com/docs/books-product/1/overview>

**Elementos a utilizar:**

1. Crea una cuenta de Developer en la página de New York Times para tener un API key (<https://developer.nytimes.com/>)
2. Usa el “Books API”.

**¿Qué debe poder hacer tu código?**

1. Pedirle al usuario decidir qué lista de “Best Sellers” quiere consultar.

import requests

def obtener\_lista\_bestsellers():

listas\_disponibles = [

"hardcover-fiction",

"hardcover-nonfiction",

"paperback-fiction",

"paperback-nonfiction",

"advice-how-to-and-miscellaneous"

# Agrega más listas según sea necesario

]

print("Selecciona una lista de 'Best Sellers' para consultar:")

for i, lista in enumerate(listas\_disponibles, start=1):

print(f"{i}. {lista.replace('-', ' ').title()}")

eleccion = int(input("Introduce el número de la lista que deseas consultar: "))

lista\_seleccionada = listas\_disponibles[eleccion - 1]

url = "https://api.nytimes.com/svc/books/v3/lists.json"

params = {

"api-key": "alex\_apy",

}

response = requests.get(url, params=params)

if response.status\_code == 200:

data = response.json()

print(data)

else:

print(f"Error: {response.status\_code}")

if \_\_name\_\_ == "\_\_main\_\_":

obtener\_lista\_bestsellers()

1. Poder escoger si quiere ver los “Best Sellers” actuales o de alguna fecha en específica.

import requests

def obtener\_lista\_bestsellers():

listas\_disponibles = [

"hardcover-fiction",

"hardcover-nonfiction",

"paperback-fiction",

"paperback-nonfiction",

"advice-how-to-and-miscellaneous"

# Agrega más listas según sea necesario

]

print("Selecciona una lista de 'Best Sellers' para consultar:")

for i, lista in enumerate(listas\_disponibles, start=1):

print(f"{i}. {lista.replace('-', ' ').title()}")

eleccion = int(input("Introduce el número de la lista que deseas consultar: "))

lista\_seleccionada = listas\_disponibles[eleccion - 1]

print("¿Quieres ver los 'Best Sellers' actuales o de una fecha específica?")

print("1. Actuales")

print("2. Fecha específica")

opcion = int(input("Introduce el número de tu elección: "))

if opcion == 1:

fecha = None

elif opcion == 2:

fecha = input("Introduce la fecha en formato YYYY-MM-DD: ")

else:

print("Opción no válida")

return

url = "https://api.nytimes.com/svc/books/v3/lists.json"

params = {

"api-key": "alex\_apy",

"list": lista\_seleccionada

}

if fecha:

params["published-date"] = fecha

response = requests.get(url, params=params)

if response.status\_code == 200:

data = response.json()

print(data)

else:

print(f"Error: {response.status\_code}")

if \_\_name\_\_ == "\_\_main\_\_":

obtener\_lista\_bestsellers()

1. Poder escoger un precio específico del libro que quiere adquirir.

import requests

def obtener\_lista\_bestsellers():

listas\_disponibles = [

"hardcover-fiction",

"hardcover-nonfiction",

"paperback-fiction",

"paperback-nonfiction",

"advice-how-to-and-miscellaneous"

# Agrega más listas según sea necesario

]

print("Selecciona una lista de 'Best Sellers' para consultar:")

for i, lista in enumerate(listas\_disponibles, start=1):

print(f"{i}. {lista.replace('-', ' ').title()}")

eleccion = int(input("Introduce el número de la lista que deseas consultar: "))

lista\_seleccionada = listas\_disponibles[eleccion - 1]

print("¿Quieres ver los 'Best Sellers' actuales o de una fecha específica?")

print("1. Actuales")

print("2. Fecha específica")

opcion = int(input("Introduce el número de tu elección: "))

if opcion == 1:

fecha = None

elif

1. Poder escoger un rango de edades dirigido para el libro.

import requests

def obtener\_lista\_bestsellers():

listas\_disponibles = [

"hardcover-fiction",

"hardcover-nonfiction",

"paperback-fiction",

"paperback-nonfiction",

"advice-how-to-and-miscellaneous"

# Agrega más listas según sea necesario

]

print("Selecciona una lista de 'Best Sellers' para consultar:")

for i, lista in enumerate(listas\_disponibles, start=1):

print

**Resultado:**

1. Poder verlo de manera estructurada, poniendo la información clave del libro que estás recomendado para la información que te dio el usuario.
2. Poder acceder rápidamente a la reseña generada por el NYT acerca del libro recomendado.
3. Para los Best Sellers actuales, decirle al usuario dónde lo puede comprar.
4. Poder mostrar toda esta información en una aplicación / pantalla / interfaz gráfica amigable para que el usuario pueda interactuar con los resultados.
5. **Examen técnico**

Explica con el mayor detalle las respuestas a estas preguntas (45 minutos).

1. La información del ERP del cliente vive en un data center controlado por un proveedor, con reglas que no permiten adicionar campos, columnas, construir tablas o extraer información. ¿Qué pasos harías para pasar de la información en el ERP, cruzar y transformarla, y mandarla a una herramienta de Business Intelligence con actualizaciones automáticas?

## Desafío de la Extracción de Datos de un ERP Restringido y Solución Propuesta

**Entendiendo el Problema:**

La principal dificultad radica en las restricciones impuestas por el proveedor del data center, que limitan las operaciones directas sobre la base de datos del ERP. Sin embargo, existen varias estrategias que podemos emplear para superar este obstáculo y lograr nuestro objetivo de alimentar una herramienta de Business Intelligence.

**Solución Propuesta:**

**Identificar el Método de Extracción Disponible:**

* + **Reportes Ad-hoc:** La mayoría de los ERP permiten generar reportes personalizados. Identifica qué tipo de reportes puedes crear y en qué formato (CSV, Excel, etc.). Estos reportes pueden ser la base para extraer los datos necesarios.
  + **Web Services:** Si el ERP expone una API o web services, puedes utilizarlos para realizar consultas programadas y obtener los datos en formato JSON o XML.
  + **ETL (Extract, Transform, Load):** Evalúa si el proveedor ofrece alguna herramienta ETL que permita extraer datos de forma programada, aunque sea con limitaciones. **Automatizar la Extracción**
  + **Scheduler:** Utiliza una herramienta de programación de tareas (como cron en Linux o Task Scheduler en Windows) para ejecutar periódicamente los scripts o herramientas que generan los reportes o consumen los web services.
  + **Herramientas de Automatización:** Considera herramientas como Python (con librerías como pandas, requests), R, o incluso herramientas de automatización visual como UiPath, para automatizar la extracción y transformación de los datos.

**Almacenamiento Intermedio:**

* + **Base de Datos Local:** Crea una base de datos local (SQL Server, PostgreSQL, etc.) donde almacenar los datos extraídos. Esto te permitirá realizar transformaciones y unificar datos de diferentes fuentes.
  + **Cloud Storage:** Plataformas como Google Cloud Storage, AWS S3 o Azure Blob Storage pueden ser una buena opción para almacenar grandes volúmenes de datos de manera segura y escalable.

1. El cliente de la pregunta 1 está evaluando adquirir un servidor físico o en la nube para poder alojar ahí un espejo de la información de su ERP. ¿Qué recomendarías en cuanto a costo o inversión y el rendimiento esperado dadas las características de cada una de las opciones?

Azure ofrece soporte y conexión de trabajar en la nube

1. Un cliente solamente cuenta con un servidor de producción donde viven los datos, ¿qué sugerirías para no afectar el desempeño de esta al conectar BI?
   1. ¿Cuáles serían las implicaciones (tiempo, conocimiento, administración, costo) de esta sugerencia?

**Data Warehouse o Data Lake:** La mejor práctica es extraer los datos del ERP a un entorno aislado, como un data warehouse o data lake. Este entorno estará dedicado exclusivamente a los análisis de BI y no afectará las operaciones diarias del ERP.

1. Un query que escribiste regresa 10,000 filas únicamente, pero su ejecución toma entre 3 y 4 horas. Por lo mismo, muchas veces no llegan los resultados completos a BI o se pierde la conexión mientras esto ocurre. ¿Qué harías?

**Reescribir la consulta:** Simplificar la lógica de la consulta, utilizar índices adecuados y evitar funciones que puedan ser costosas.

**Query optimizers:** Utilizar herramientas especializadas para analizar y optimizar consultas SQL.

1. Te busca un cliente para decirte que los resultados que ve en los tableros no le hacen sentido (ayer si mostraba resultados correctos). ¿Qué pasos harías para darle una respuesta al cliente?

Revisar fuente de datos y que tub¿viese actualizacion optima del tablero

1. ¿Qué mejores prácticas sugerirías para garantizar la seguridad de la información del cliente, ya sea en un servidor físico o en la nube?

**Seguridad compartida:** Entender que la seguridad en la nube es una responsabilidad compartida entre el proveedor de la nube y el cliente.

* **Configuración segura:** Asegurarse de que la configuración de los servicios en la nube sea segura y cumpla con las mejores prácticas.
* **Monitoreo continuo:** Implementar herramientas de monitoreo para detectar anomalías y responder rápidamente a cualquier amenaza.

1. El director comercial de una empresa nos pide un proyecto para poder visualizar información en un tablero de BI con actualizaciones cada 15 minutos con información proveniente de distintas fuentes:
   1. ERP (SQL Server)
   2. CRM (API)
   3. Catálogos de información (Excel)
   4. Google Analytics
   5. Google Ads
   6. Shopify
   7. Facebook Ads

Prepara una propuesta técnica que incluya tiempos, costos y herramientas a utilizar para entregarle al director comercial este tablero. En caso de que haya información que necesites para hacer esta propuesta y no la tengas, genera supuestos para los que sería valida la propuesta.

### Herramientas Propuestas

* **Herramienta de BI:** Power BI (por su amplia gama de funcionalidades, facilidad de uso y integración con diversas fuentes de datos).
* **Herramientas de ETL:** Azure Data Factory o SSIS para la extracción, transformación y carga de datos.
* **Herramientas de Versionamiento:** Git para el control de versiones del código.
* **Herramientas de Colaboración:** Microsoft Teams o Slack para la comunicación y coordinación del equipo.