In [1]:
```
# Importar librerías
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import random
```

In [2]:
```
# Cargar las bases de datos desde los archivos CSV en GitHub
client_url = "https://raw.githubusercontent.com/annaalfaro/TFM/main/Client.csv"
supplier_url = "https://raw.githubusercontent.com/annaalfaro/TFM/main/Supplier.csv"
customer_item_url = "https://raw.githubusercontent.com/annaalfaro/TFM/main/CustomerItem.csv"
item_url = "https://raw.githubusercontent.com/annaalfaro/TFM/main/Item.csv"



# Leer los datos con codificación ISO-8859-1
clients = pd.read_csv(client_url, sep=';', encoding='ISO-8859-1')
suppliers = pd.read_csv(supplier_url, sep=';', encoding='ISO-8859-1')
customer_items = pd.read_csv(customer_item_url, sep=';', encoding='ISO-8859-1')
items = pd.read_csv(item_url, sep=';', encoding='ISO-8859-1')
```

In [3]:
```
# Crear la base de datos SalesOrder
sales_order_data = []

# Parámetros para la generación de datos
num_sales_orders = 400  # Número de órdenes de venta
num_lines_per_order = 5  # Máximo de líneas por orden de venta
start_date = datetime(2020, 1, 1)
end_date = datetime(2024, 7, 31)

# Inicializar una tendencia de crecimiento gradual para cada item
item_trends = {item_id: random.uniform(0.01, 0.03) for item_id in customer_items['CustomerItem_id'].unique()}

# Función para aplicar un pico de demanda pequeño (incremento de 5%)
def apply_demand_peak(base_quantity, peak_factor=1.05):
    if random.random() < 0.1:  # 10% de probabilidad de un pequeño pico
        return int(base_quantity * peak_factor)
    return base_quantity

# Simular un crecimiento en las ventas del 5% anual
annual_growth_rate = 0.05

# Encontrar el primer miércoles después de la fecha de inicio
current_date = start_date
if current_date.weekday() != 2:  # Si no es miércoles, ajusta la fecha al próximo miércoles
    days_until_wednesday = (2 - current_date.weekday() + 7) % 7
    current_date += timedelta(days=days_until_wednesday)
```

In [4]:
```
# Generación de órdenes de venta
while current_date <= end_date:
    so_id = f"SO{len(sales_order_data) // num_lines_per_order + 1:06d}"
    client_id = random.choice(customer_items['CustomerItem_client'].unique())
    client_items = customer_items[customer_items['CustomerItem_client'] == client_id]

    # Ajustar la cantidad según la estacionalidad, crecimiento y tendencia general
    year_diff = (current_date.year - start_date.year)
    growth_adjustment = (1 + annual_growth_rate) ** year_diff

    # Generar la fecha de la orden y la fecha estimada de entrega solo una vez por SO
    so_date = current_date.strftime("%Y-%m-%d")
    so_est_date = (current_date + timedelta(days=random.randint(15, 60))).strftime("%Y-%m-%d")

    # Generar las líneas de la orden
    num_lines = random.randint(1, min(num_lines_per_order, len(client_items)))  # Limitar las líneas al número de artículos únicos disp
    chosen_items = set()  # Para controlar que no se repitan artículos en la misma orden

    for ln in range(1, num_lines + 1):
        # Escoger un artículo que no haya sido ya escogido en esta orden
        available_items = list(set(client_items['CustomerItem_id'].unique()) - chosen_items)
        if not available_items:
            break  # Si ya no hay artículos disponibles, termina el bucle

        so_ln = ln
        so_customer_item_id = random.choice(available_items)
        chosen_items.add(so_customer_item_id)

        # Obtener la tendencia específica del ítem
        item_trend = item_trends[so_customer_item_id]

        # Base quantity con un incremento gradual y variación aleatoria
        base_quantity = int(100 * (1 + item_trend * year_diff) * growth_adjustment)

        # Aplica un pequeño incremento ocasional
        so_quantity = apply_demand_peak(base_quantity)

        # Añadir una pequeña variación aleatoria
        so_quantity = so_quantity + random.randint(-10, 10)
        so_quantity = max(1, so_quantity)  # Evitar cantidades negativas

        # Añadir la línea de la orden de venta
        sales_order_data.append({
            "SO_id": so_id,
            "SO_ln": so_ln,
            "SO_CustomerItemid": so_customer_item_id,
            "SO_Quantity": so_quantity,
            "SO_Client": client_id,  # Asegurar que todas las líneas tienen el mismo cliente
            "SO_Date": so_date,  # Fecha única para la orden
            "SO_EstDate": so_est_date  # Fecha estimada de entrega única para la orden
        })

    # Avanzar al siguiente miércoles
    current_date += timedelta(weeks=1)
```

In [5]:
```
# Convertir la lista a un DataFrame
sales_order_df = pd.DataFrame(sales_order_data)

# Formato fecha de tipo datetime
sales_order_df['SO_EstDate'] = pd.to_datetime(sales_order_df['SO_EstDate'])

# Ordenar por SO_EstDate y luego por SO_ln
sales_order_df_sorted = sales_order_df.sort_values(by=['SO_EstDate', 'SO_id', 'SO_ln'])

# Mostrar las primeras filas ordenadas
sales_order_df_sorted.head(20)
```

Out[5]:

	SO_id	SO_ln	SO_CustomerItemid	SO_Quantity	SO_Client	SO_Date	SO_EstDate
0	SO000001	1	CI006	92	CL001	2020-01-01	2020-01-16
1	SO000001	2	CI001	103	CL001	2020-01-01	2020-01-16
2	SO000001	3	CI002	101	CL001	2020-01-01	2020-01-16
3	SO000001	1	CI006	104	CL001	2020-01-08	2020-02-04
4	SO000001	2	CI007	103	CL001	2020-01-08	2020-02-04
5	SO000001	3	CI005	109	CL001	2020-01-08	2020-02-04
6	SO000002	1	CI004	104	CL005	2020-01-15	2020-03-08
7	SO000002	1	CI010	92	CL004	2020-01-22	2020-03-09
8	SO000002	1	CI003	104	CL002	2020-01-29	2020-03-26
9	SO000002	1	CI005	100	CL001	2020-02-05	2020-03-31
10	SO000002	2	CI006	106	CL001	2020-02-05	2020-03-31
11	SO000002	3	CI002	94	CL001	2020-02-05	2020-03-31
12	SO000002	4	CI007	97	CL001	2020-02-05	2020-03-31
15	SO000004	1	CI002	92	CL001	2020-02-26	2020-03-31
16	SO000004	2	CI005	99	CL001	2020-02-26	2020-03-31
17	SO000004	3	CI006	102	CL001	2020-02-26	2020-03-31
18	SO000004	4	CI007	101	CL001	2020-02-26	2020-03-31
19	SO000004	5	CI001	94	CL001	2020-02-26	2020-03-31
13	SO000003	1	CI010	93	CL004	2020-02-12	2020-04-01
14	SO000003	1	CI007	105	CL001	2020-02-19	2020-04-07

In [6]:
```
# Guardar como un archivo CSV en la carpeta local
csv_filename = "SalesOrder.csv"
sales_order_df_sorted.to_csv(csv_filename, index=False)
```

In [ ]:

In [ ]: