GENERACIÓN DE ORDENES DE COMPRA A PROVEEDORES A PARTIR DE LAS ORDENES DE VENTA

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

Primero se cargan las bases de datos y se leen con formato ISO y con el separador especificado.

In [8]:
```
# Cargar las bases de datos desde los archivos CSV en GitHub
customer_item_url = "https://raw.githubusercontent.com/annaalfaro/TFM/main/CustomerItem.csv"
item_url = "https://raw.githubusercontent.com/annaalfaro/TFM/main/Item.csv"
supplier_url = "https://raw.githubusercontent.com/annaalfaro/TFM/main/Supplier.csv"
sales_order_url = "https://raw.githubusercontent.com/annaalfaro/TFM/main/prediccionesSO.csv"

# Leer los datos con codificación 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')
suppliers = pd.read_csv(supplier_url, sep=';', encoding='ISO-8859-1')
sales_order = pd.read_csv(sales_order_url, sep=',', encoding='ISO-8859-1')
```

In [9]:
```
# Crear un diccionario de mapeo para los valores de 0 a 9 a CI001 a CI010
mapping = {i: f"CI{i+1:03d}" for i in range(10)}

# Aplicar el mapeo a la columna 'SO_CustomerItemid'
sales_order['SO_CustomerItemid'] = sales_order['SO_CustomerItemid'].map(mapping).fillna(sales_order['SO_CustomerItemid'])
```

En el siguiente codigo se prepara para la generación de la base de datos para las ordenes de compra que se envían al proveedor a partir de las ordenes de ventas que se generaron previamente del cliente. Para ello se han precisado las bases de datos creadas manualmente y las creadas con el otro codigo, además de parametros externos como los días de envío y unos días de margen por si suceden algunos problemas.

In [10]:
```
# Comprobar que las columnas de fecha sean de tipo datetime
sales_order['SO_Date'] = pd.to_datetime(sales_order['SO_Date'])
sales_order['SO_EstDate'] = pd.to_datetime(sales_order['SO_EstDate'])

# Parámetros externos
empresa_envio_dias = 3  # Días que tarda nuestra empresa en enviar el producto
margen_problemas_dias = 5  # Margen adicional para problemas

# Crear la base de datos Purchase_Order
purchase_order_data = []

# Agrupar por SO_CustomerItemid y Supplier para generar las PO
for customer_item_id, group in sales_order.groupby('SO_CustomerItemid'):
    # Subdividir los grupos por Supplier
    def get_supplier(x):
        matching_items = items[items['Item_FinalItem'] == group.loc[x, 'SO_CustomerItemid']]
        if not matching_items.empty:
            return matching_items.iloc[0]['Item_Supplier']
        else:
            # Manejo de error si no se encuentra el proveedor
            raise ValueError(f"No se encontró el proveedor para el CustomerItemid {group.loc[x, 'SO_CustomerItemid']}")

    # Agrupar por el resultado de la función get_supplier
    subgroups = group.groupby(get_supplier)

    for supplier_id, subgroup in subgroups:
        # Crear un identificador de PO
        po_id = f"PO{len(purchase_order_data) + 1:06d}"
        po_ln_counter = 1  # Reiniciar el contador de PO_ln para cada nuevo PO_id

        # Inicializar variables para calcular cantidades y fechas
        consolidated_items = {}
        max_so_est_date = pd.to_datetime(subgroup['SO_EstDate'].max())
        min_po_date = max_so_est_date

        # Procesar cada artículo en el grupo
        for _, so_row in subgroup.iterrows():
            item_row = items[items['Item_FinalItem'] == so_row['SO_CustomerItemid']].iloc[0]
            item_id = item_row['Item_id']
            po_prodtime = item_row['Item_ProdTime']
            supplier_transport_days = suppliers[suppliers['Supplier_id'] == supplier_id].iloc[0]['Supplier_TransportDays']

            # Calcular la fecha estimada de la PO y la fecha de la PO
            total_days = int(po_prodtime + supplier_transport_days + empresa_envio_dias + margen_problemas_dias)
            po_est_date = max_so_est_date - timedelta(days=total_days)
            min_po_date = min(min_po_date, po_est_date - timedelta(days=random.randint(1, 10)))

            # Consolidar cantidades
            if item_id in consolidated_items:
                consolidated_items[item_id]['PO_Quantity'] += so_row['Predicted_Quantity']
            else:
                consolidated_items[item_id] = {
                    'PO_Quantity': so_row['Predicted_Quantity'],
                    'PO_estDate': po_est_date
                }

        # Crear entradas en Purchase_Order para cada artículo consolidado
        for item_id, details in consolidated_items.items():
            purchase_order_data.append({
                "PO_id": po_id,
                "PO_ln": po_ln_counter,
                "PO_item": item_id,
                "PO_Quantity": details['PO_Quantity'],
                "PO_Supplier": supplier_id,
                "PO_date": min_po_date.strftime("%Y-%m-%d"),
                "PO_estDate": details['PO_estDate'].strftime("%Y-%m-%d"),
                "PO_CustomerItemId": customer_item_id,
                "PO_SOEstDate": max_so_est_date.strftime("%Y-%m-%d")  # Añadir la fecha estimada de la Sales Order
            })
            po_ln_counter += 1
```

En el siguiente código se consolida y ordena la base de datos.

In [11]:
```
# Convertir la lista a un DataFrame
purchase_order_df = pd.DataFrame(purchase_order_data)

# Ordenar por PO_id y PO_ln
purchase_order_df_sorted = purchase_order_df.sort_values(by=['PO_id', 'PO_ln'])

# Mostrar las primeras filas del DataFrame ordenado
purchase_order_df_sorted.head(20)
```

Out[11]:

	PO_id	PO_ln	PO_item	PO_Quantity	PO_Supplier	PO_date	PO_estDate	PO_CustomerItemId	PO_SOEstDate
0	PO000001	1	IT001	3232.722949	SU001	2025-07-21	2025-07-31	CI001	2025-08-14
1	PO000002	1	IT002	5551.116498	SU002	2025-10-01	2025-10-11	CI002	2025-10-25
2	PO000003	1	IT003	5667.154830	SU003	2025-09-19	2025-09-29	CI003	2025-10-13
3	PO000004	1	IT004	5143.605700	SU004	2025-09-22	2025-10-02	CI004	2025-10-14
4	PO000005	1	IT005	5011.575574	SU001	2025-09-15	2025-09-25	CI005	2025-10-12
5	PO000006	1	IT006	5405.463010	SU005	2025-09-06	2025-09-16	CI006	2025-09-29
6	PO000007	1	IT007	4480.068792	SU006	2025-09-25	2025-10-05	CI007	2025-10-18
7	PO000008	1	IT008	3689.358879	SU007	2025-10-10	2025-10-20	CI008	2025-11-01
8	PO000009	1	IT009	4942.676231	SU008	2025-10-01	2025-10-11	CI009	2025-10-28
9	PO000010	1	IT010	4754.863149	SU002	2025-09-15	2025-09-25	CI010	2025-10-13

Finalmente se guarda en un archivo csv en local para luego enviarlo a GITHUB

In [12]:
```
# Guardar el DataFrame ordenado como un archivo CSV
csv_filename = "PurchaseOrderPREDICTED.csv"
purchase_order_df_sorted.to_csv(csv_filename, index=False)
```

In [ ]:
```
```