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 [2]:
```
# 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/SalesOrder.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')
```

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 [3]:
```
# 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_id y Supplier para generar las PO
for so_id, group in sales_order.groupby('SO_id'):
    # Subdividir los grupos por Supplier
    subgroups = group.groupby(lambda x: items[items['Item_FinalItem'] == group.loc[x, 'SO_CustomerItemid']].iloc[0]['Item_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['SO_Quantity']
                consolidated_items[item_id]['PO_SOln'].append(so_row['SO_ln'])
            else:
                consolidated_items[item_id] = {
                    'PO_Quantity': so_row['SO_Quantity'],
                    'PO_SOln': [so_row['SO_ln']],
                    '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_SO": so_id,
                "PO_SOln": ", ".join(map(str, details['PO_SOln'])),
                "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 [4]:
```
# 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[4]:

	PO_id	PO_ln	PO_item	PO_Quantity	PO_Supplier	PO_date	PO_estDate	PO_SO	PO_SOln	PO_SOEstDate
0	PO000001	1	IT001	103	SU001	2020-01-14	2020-01-21	SO000001	2	2020-02-04
1	PO000001	2	IT005	109	SU001	2020-01-14	2020-01-18	SO000001	3	2020-02-04
2	PO000003	1	IT002	101	SU002	2019-12-29	2020-01-02	SO000001	3	2020-01-16
3	PO000004	1	IT006	196	SU005	2020-01-14	2020-01-22	SO000001	1, 1	2020-02-04
4	PO000005	1	IT007	103	SU006	2020-01-16	2020-01-22	SO000001	2	2020-02-04
5	PO000006	1	IT005	100	SU001	2020-03-13	2020-03-14	SO000002	1	2020-03-31
6	PO000007	1	IT010	92	SU002	2020-03-08	2020-03-13	SO000002	1	2020-03-31
7	PO000007	2	IT002	94	SU002	2020-03-08	2020-03-17	SO000002	3	2020-03-31
8	PO000009	1	IT003	104	SU003	2020-03-08	2020-03-12	SO000002	1	2020-03-26
9	PO000010	1	IT004	104	SU004	2020-02-17	2020-02-25	SO000002	1	2020-03-08
10	PO000011	1	IT006	106	SU005	2020-03-16	2020-03-18	SO000002	2	2020-03-31
11	PO000012	1	IT007	97	SU006	2020-03-17	2020-03-18	SO000002	4	2020-03-31
12	PO000013	1	IT010	93	SU002	2020-03-07	2020-03-14	SO000003	1	2020-04-01
13	PO000014	1	IT007	105	SU006	2020-03-20	2020-03-25	SO000003	1	2020-04-07
14	PO000015	1	IT005	99	SU001	2020-03-06	2020-03-14	SO000004	2	2020-03-31
15	PO000015	2	IT001	94	SU001	2020-03-06	2020-03-17	SO000004	5	2020-03-31
16	PO000017	1	IT002	92	SU002	2020-03-15	2020-03-17	SO000004	1	2020-03-31
17	PO000018	1	IT006	102	SU005	2020-03-10	2020-03-18	SO000004	3	2020-03-31
18	PO000019	1	IT007	101	SU006	2020-03-17	2020-03-18	SO000004	4	2020-03-31
19	PO000020	1	IT005	100	SU001	2020-03-27	2020-03-28	SO000005	1	2020-04-14

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

In [5]:
```
# Guardar el DataFrame ordenado como un archivo CSV
csv_filename = "PurchaseOrder.csv"
purchase_order_df_sorted.to_csv(csv_filename, index=False)
```

In [ ]: