In [1]:
```
import sqlite3
import pandas as pd

# Conexión a SQLite
conn = sqlite3.connect("Retail.db")
cur = conn.cursor()
```

In [3]:
```
# Tabla Customers
cur.execute("""
CREATE TABLE IF NOT EXISTS Customers (
    CustomerID TEXT PRIMARY KEY,
    CustomerName TEXT,
    Segment TEXT,
    Country TEXT,
    City TEXT,
    State TEXT,
    PostalCode TEXT
)
""")

# Tabla Products
cur.execute("""
CREATE TABLE IF NOT EXISTS Products (
    ProductID TEXT PRIMARY KEY,
    ProductName TEXT,
    Category TEXT,
    SubCategory TEXT,
    Price REAL
)
""")

# Tabla Orders
cur.execute("""
CREATE TABLE IF NOT EXISTS Orders (
    OrderID TEXT PRIMARY KEY,
    CustomerID TEXT,
    ProductID TEXT,
    OrderDate TEXT,
    ShipDate TEXT,
    ShipMode TEXT,
    Quantity INTEGER,
    Discount REAL,
    Profit REAL,
    Sales REAL,
    Region TEXT,
    FOREIGN KEY(CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY(ProductID) REFERENCES Products(ProductID)
)
""")

conn.commit()
```

In [5]:
```
import os
import csv

os.chdir("/Users/Lenovo/Desktop/EBAC/SQL")
customers_df = pd.read_csv("Customers.csv")
products_df = pd.read_csv("Products.csv")
orders_df = pd.read_csv("Orders.csv")


customers_df.to_sql("Customers", conn, if_exists="append", index=False)
products_df.to_sql("Products", conn, if_exists="append", index=False)
orders_df.to_sql("Orders", conn, if_exists="append", index=False)
```

Out[5]: 500

In [7]:
```
# Ver solo los pedidos que existen
query_inner = """
SELECT o.OrderID, c.CustomerName, p.ProductName, o.Sales, o.Profit
FROM Orders o
INNER JOIN Customers c ON o.CustomerID = c.CustomerID
INNER JOIN Products p ON o.ProductID = p.ProductID
LIMIT 10
"""
pd.read_sql(query_inner, conn)
```

Out[7]:

	OrderID	CustomerName	ProductName	Sales	Profit
0	O001	Judith Mendoza	Avoid Provide	4284.65	696.29
1	O002	Michael Morris	Determine Full	678.25	200.80
2	O003	Robin Hill	Scientist Consider	8361.62	2287.05
3	O004	Amber Morrison	High Occur	10724.96	2995.73
4	O005	Sarah Garcia	Contain Red	3070.53	561.23
5	O006	Billy Jackson	Popular Choose	1584.43	144.58
6	O007	Ashley Holland	Quality Watch	325.91	87.87
7	O008	Jesse Cuevas	Impact Past	14198.22	3677.93
8	O009	Sheila Harris	Fill Kid	4192.34	554.48
9	O010	Sarah Boone	Run They	2855.97	429.93

In [9]:
```
# Listar todos los clientes, aunque algunos no hayan realizado pedidos.
query_left = """
SELECT c.CustomerName, o.OrderID, o.Sales
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
LIMIT 10
"""
pd.read_sql(query_left, conn)
```

Out[9]:

	CustomerName	OrderID	Sales
0	Alexandra Berger	O207	14542.90
1	Alexandra Berger	O358	6557.37
2	Peter Sanders	None	NaN
3	Sue Cooper	O364	2342.92
4	Tara Davis	O486	4278.51
5	Tracy Craig	None	NaN
6	Sheila Harris	O009	4192.34
7	Christina Burke	None	NaN
8	Matthew Dyer	O389	13236.03
9	Terry Sanchez	O037	996.95

In [11]:
```
# Categorizar los pedidos según el monto de venta.
query_case = """
SELECT OrderID, Sales,
CASE
    WHEN Sales > 1000 THEN 'Alta'
    WHEN Sales BETWEEN 500 AND 1000 THEN 'Media'
    ELSE 'Baja'
END AS Categoria_Ventas
FROM Orders
LIMIT 10
"""
pd.read_sql(query_case, conn)
```

Out[11]:

	OrderID	Sales	Categoria_Ventas
0	O001	4284.65	Alta
1	O002	678.25	Media
2	O003	8361.62	Alta
3	O004	10724.96	Alta
4	O005	3070.53	Alta
5	O006	1584.43	Alta
6	O007	325.91	Baja
7	O008	14198.22	Alta
8	O009	4192.34	Alta
9	O010	2855.97	Alta

In [13]:
```
# Identificar solo los clientes más importantes, con ventas acumuladas altas.
query_sub = """
```

```
SELECT CustomerName
FROM Customers c
WHERE (SELECT SUM(Sales) FROM Orders o WHERE o.CustomerID = c.CustomerID) > 3000
LIMIT 10
"""
pd.read_sql(query_sub, conn)
```

Out[13]:

	CustomerName
0	Alexandra Berger
1	Tara Davis
2	Sheila Harris
3	Matthew Dyer
4	Jesse Cuevas
5	James Price
6	Karen Hayes
7	Dominic Thomas
8	William Glenn
9	Russell Alvarez

In [15]:
```
# Qué productos han comprado clientes de segmento Corporate.
query_semijoin = """
SELECT DISTINCT ProductID, ProductName
FROM Products p
WHERE ProductID IN (
    SELECT ProductID FROM Orders o
    JOIN Customers c ON o.CustomerID = c.CustomerID
    WHERE c.Segment = 'Corporate'
)
LIMIT 10
"""
pd.read_sql(query_semijoin, conn)
```

Out[15]:

	ProductID	ProductName
0	P002	Wall Performance
1	P003	Now Age
2	P004	Measure Still
3	P007	Full Until
4	P008	Speak Assume
5	P010	Or Issue
6	P013	Business Fill
7	P021	Enter Above
8	P022	Game Stop
9	P023	Policy Occur

In [17]:
```
# Identificar clientes que nunca compraron productos de la categoría Technology.
query_antijoin = """
SELECT CustomerName
FROM Customers
WHERE CustomerID NOT IN (
    SELECT o.CustomerID
    FROM Orders o
    JOIN Products p ON o.ProductID = p.ProductID
    WHERE p.Category = 'Technology'
)
LIMIT 10
"""
pd.read_sql(query_antijoin, conn)
```

Out[17]:

	CustomerName
0	Peter Sanders
1	Sue Cooper
2	Tara Davis
3	Tracy Craig
4	Christina Burke
5	Terry Sanchez
6	Jesse Cuevas
7	William Glenn
8	Edwin Haynes
9	Russell Alvarez

In [19]:
```
# Calcular el total de ventas de cada cliente.
query_sub_select = """
SELECT CustomerName,
(SELECT SUM(Sales) FROM Orders o WHERE o.CustomerID=c.CustomerID) AS TotalSales
FROM Customers c
LIMIT 10
"""
pd.read_sql(query_sub_select, conn)
```

Out[19]:

	CustomerName	TotalSales
0	Alexandra Berger	21100.27
1	Peter Sanders	NaN
2	Sue Cooper	2342.92
3	Tara Davis	4278.51
4	Tracy Craig	NaN
5	Sheila Harris	4192.34
6	Christina Burke	NaN
7	Matthew Dyer	13236.03
8	Terry Sanchez	996.95
9	Jesse Cuevas	20920.94

In [21]:
```
conn.close()
```

Para este ejercicio cree aleatoriamente tres bases de datos simulando ventas en Retail, con este hice varias relaciones entre clientes, categorias de productos, segmentos y productos.

In [ ]: