In [2]:
```
import pandas as pd
```

In [8]:
```
order = pd.read_excel("/Users/vignesh/Documents/george brown pgdm /Foundation of data management/Lab Exercises
```

In [11]:
```
order.head()
```

Out[11]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country/Region	City	...	Postal Code	Region	Produc I
0	1	CA-2020-152156	2020-11-08	2020-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420.0	South	FUR-BO 1000179
1	2	CA-2020-152156	2020-11-08	2020-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420.0	South	FUR-CH 1000045
2	3	CA-2020-138688	2020-06-12	2020-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036.0	West	OFF-LA 1000024
3	4	US-2019-108966	2019-10-11	2019-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311.0	South	FUR-TA 1000057
4	5	US-2019-108966	2019-10-11	2019-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311.0	South	OFF-ST 1000076

5 rows × 21 columns

Add columns to source data based on assumptions

In [38]:
```
order["Profit Margin"] = order["Profit"] / order["Sales"]
```

In [41]:
```
order["Cost of Goods Sold"] = (((1-(order["Profit Margin"]-order["Discount"]))/3)*2)*order["Sales"]
```

In [44]:
```
order["Cost of Sales"] = (((1-(order["Profit Margin"]-order["Discount"]))/3)/2)*order["Sales"]
```

In [45]:
```
order["Other Expenses"] = (((1-(order["Profit Margin"]-order["Discount"]))/3)/2)*order["Sales"]
```

In [46]:
```
order.columns
```

Out[46]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country/Region', 'City',
       'State', 'Postal Code', 'Region', 'Product ID', 'Category',
       'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',
       'Profit', 'Profit Margin', 'Cost of Goods Sold', 'Cost of Sales',
       'Other Expenses'],
      dtype='object')
```

Split data into different entities

Order table		Product table		Region table
Order ID: PK		Product ID: PK		Region ID: PK
Order Date		Category		Region
Ship Date		Sub-Category		Regional Manager
Ship Mode		Product Name		
Region ID: FK				
Customer ID: FK				
Order item table		**Customer table**		**Return table**
Order item ID: PK		Customer ID: PK		Return ID: PK
Product ID: FK		Customer Name		Order ID: FK
Order ID: FK				
Sales		Segment		
Quantity		Country/Region		
Discount		City		
Profit		State		
Profit Margin		Postal Code		
Cost of Goods Sold (COGS)				
Cost of Sales				
Other Expenses				

One to many: One product can be there in multiple order items

In [70]:
```
product = order[['Product ID', 'Category', 'Sub-Category', 'Product Name']].groupby("Product ID", as_index=Fa
```

In [274…
```
product.head()
```

Out[274]:

	Product ID	Category	Sub-Category	Product Name
0	FUR-BO-10000112	Furniture	Bookcases	Bush Birmingham Collection Bookcase, Dark Cherry
1	FUR-BO-10000330	Furniture	Bookcases	Sauder Camden County Barrister Bookcase, Plank...
2	FUR-BO-10000362	Furniture	Bookcases	Sauder Inglewood Library Bookcases
3	FUR-BO-10000468	Furniture	Bookcases	O'Sullivan 2-Shelf Heavy-Duty Bookcases
4	FUR-BO-10000711	Furniture	Bookcases	Hon Metal Bookcases, Gray

Many to one relationship: Many order can belong to one region

In [235…
```
region = pd.read_excel("/Users/vignesh/Documents/george brown pgdm /Foundation of data management/Lab Exercise
```

In [236…
```
region.reset_index(names="Region ID", inplace=True)
region = region[["Region ID", "Region", "Regional Manager"]]
```

In [237…
```
region.head()
```

Out[237]:

	Region ID	Region	Regional Manager
0	0	West	Sadie Pawthorne
1	1	East	Chuck Magee
2	2	Central	Roxanne Rodriguez
3	3	South	Fred Suzuki

One to Many relationship: One order will have multiple order item table. In other words, Multiple order item will be linked to one order table

In [242…
```
order_item = order[['Sales', 'Quantity', 'Discount','Profit', 'Profit Margin', 'Cost of Goods Sold', 'Cost of
```

In [ ]:
```
for i, row in order_item.iterrows():
    order_item.loc[i, "Order Item ID"] = row["Order ID"] + "_" +str(row["Row ID"])

order_item.drop(["Row ID"], axis=1, inplace=True)

order_item = order_item[["Order Item ID", "Order ID", "Product ID", 'Sales', 'Quantity', 'Discount', 'Profit'
       'Cost of Goods Sold', 'Cost of Sales', 'Other Expenses']]
```

In [244…
```
order_item.head()
```

Out[244]:

	Order Item ID	Order ID	Product ID	Sales	Quantity	Discount	Profit	Profit Margin	Cost of Goods Sold	Cost of Sales	Other Expenses
0	CA-2020-152156_1	CA-2020-152156	FUR-BO-10001798	261.9600	2	0.00	41.9136	0.1600	146.697600	36.674400	36.674400
1	CA-2020-152156_2	CA-2020-152156	FUR-CH-10000454	731.9400	3	0.00	219.5820	0.3000	341.572000	85.393000	85.393000
2	CA-2020-138688_3	CA-2020-138688	OFF-LA-10000240	14.6200	2	0.00	6.8714	0.4700	5.165733	1.291433	1.291433
3	US-2019-108966_4	US-2019-108966	FUR-TA-10000577	957.5775	5	0.45	-383.0310	-0.4000	1181.012250	295.253062	295.253062
4	US-2019-108966_5	US-2019-108966	OFF-ST-10000760	22.3680	2	0.20	2.5164	0.1125	16.216800	4.054200	4.054200

In [ ]:
```
from unidecode import unidecode

customer_cp = pd.DataFrame()
for i, row in customer.iterrows():
    for column in customer.columns:
        customer_cp.loc[i, column] = unidecode(row[column])
```

One to many relationship: One customer can have multiple orders

In [259…
```
customer = order[['Customer ID', 'Customer Name', 'Segment', 'Country/Region', 'City',
       'State', 'Postal Code']].groupby("Customer ID", as_index=False).first()
```

In [260…
```
customer.head()
```

Out[260]:

	Customer ID	Customer Name	Segment	Country/Region	City	State	Postal Code
0	AA-10315	Alex Avila	Consumer	United States	Minneapolis	Minnesota	55407.0
1	AA-10375	Allen Armold	Consumer	United States	Mesa	Arizona	85204.0
2	AA-10480	Andrew Allen	Consumer	United States	Concord	North Carolina	28027.0
3	AA-10645	Anna Andreadi	Consumer	United States	Chester	Pennsylvania	19013.0
4	AB-10015	Aaron Bergman	Consumer	United States	Seattle	Washington	98103.0

One to one relationship: Only one return is possible per order

We are assuming return is for order instead of order item.

In [247…
```
returns = pd.read_excel("/Users/vignesh/Documents/george brown pgdm /Foundation of data management/Lab Exercis
```

In [248…
```
returns = returns.groupby("Order ID", as_index=False).first()
```

In [249…
```
returns["Returns ID"] = returns["Order ID"].apply(lambda x: f"R_{x}")
returns.drop(["Returned"], axis=1, inplace=True)
returns = returns[["Returns ID", "Order ID"]]
```

In [250…
```
returns.head()
```

Out[250]:

	Returns ID	Order ID
0	R_CA-2018-100762	CA-2018-100762
1	R_CA-2018-100867	CA-2018-100867
2	R_CA-2018-102652	CA-2018-102652
3	R_CA-2018-103373	CA-2018-103373
4	R_CA-2018-103744	CA-2018-103744

Order table

In [233…
```
order_table = order[['Order ID', 'Order Date', 'Ship Date', 'Ship Mode', "Region", "Customer ID"]]
```

In [234…
```
order_table = order_table.groupby('Order ID', as_index=False).first()
```

In [238…
```
region_dict = dict(zip(region.Region, region["Region ID"]))

for i, row in order_table.iterrows():
    order_table.loc[i, "Region"] = region_dict.get(row["Region"])
```

In [239…
```
order_table.head()
```

Out[239]:

	Order ID	Order Date	Ship Date	Ship Mode	Region	Customer ID
0	CA-2018-100006	2018-09-07	2018-09-13	Standard Class	1	DK-13375
1	CA-2018-100090	2018-07-08	2018-07-12	Standard Class	0	EB-13705
2	CA-2018-100293	2018-03-14	2018-03-18	Standard Class	3	NF-18475
3	CA-2018-100328	2018-01-28	2018-02-03	Standard Class	1	JC-15340
4	CA-2018-100363	2018-04-08	2018-04-15	Standard Class	0	JM-15655

Write to csv

In [246…
```
product.to_csv("Product table1.csv", index=False)
region.to_csv("Region table1.csv", index=False)
order_detail.to_csv("Order Detail table1.csv", index=False)
customer.to_csv("Customer table1.csv", index=False)
returns.to_csv("Returns table1.csv", index=False)
order_table.to_csv("Order table1.csv", index=False)
```

In [270…