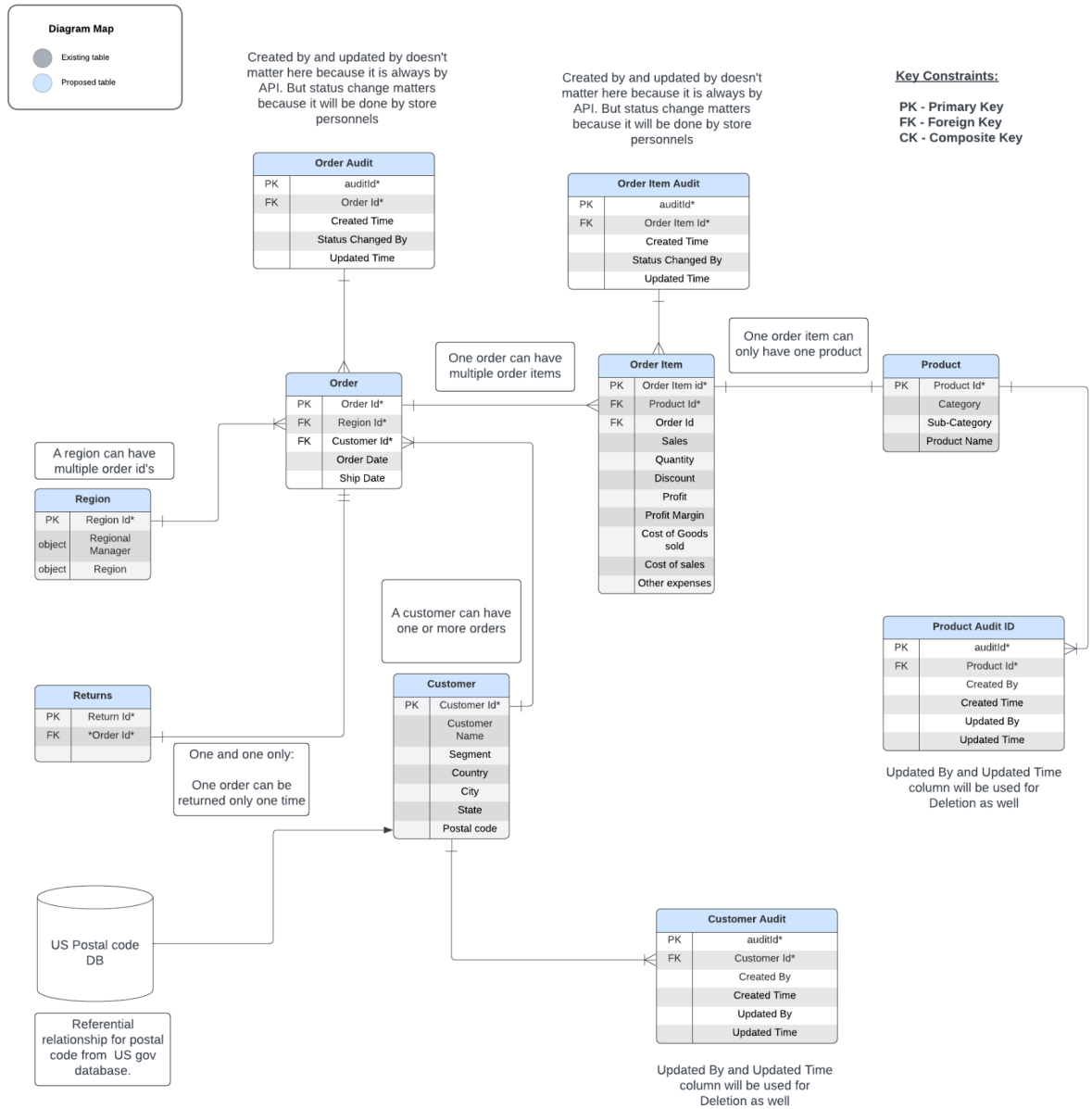
Observations: ERD: Postal Code is indeed reference data. You can create a new table for that, called “Postal Code (reference data) “ so you can explain the cardinality. Be careful not to include redundancies in Order Item. Any attribute that results on a calculation from other attributes in the same entity. it is best practice to indicate those entities with critical data, thus which ones are considered a Master Data Table. You can do so by adding to the entities name: “ (master data)” label.

Foundation of Data Management - Assignment No. 2
GROUP NO. 8

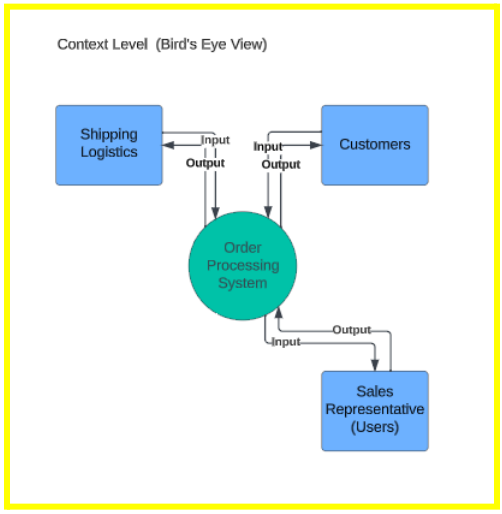
1. Logical-level ERD

Database ER Diagram (crow foot)

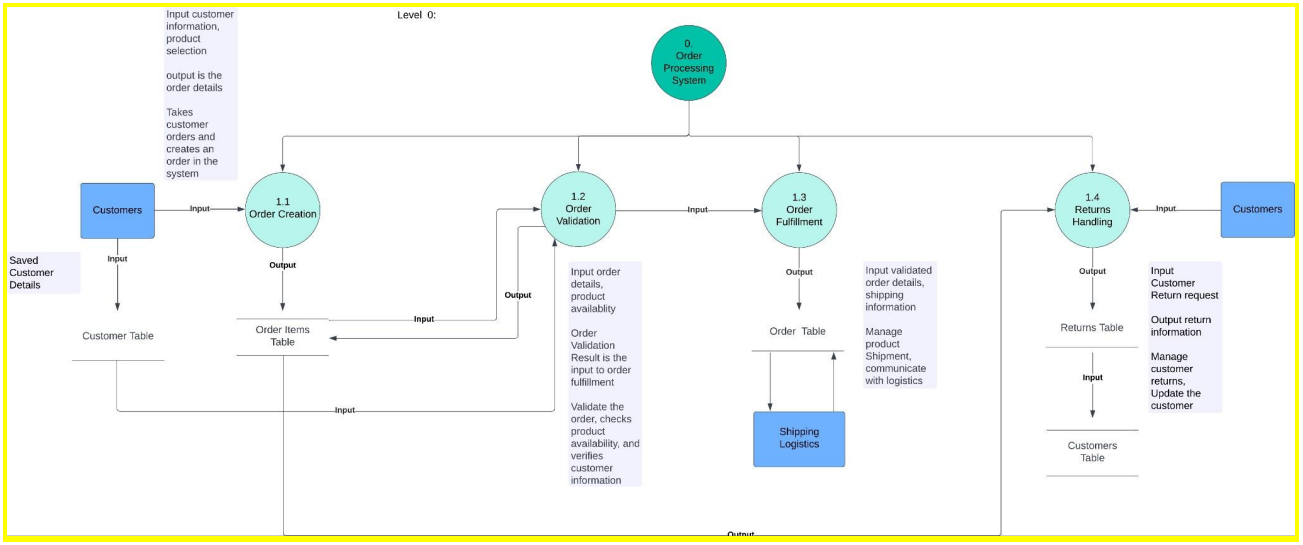


2. Data Flow

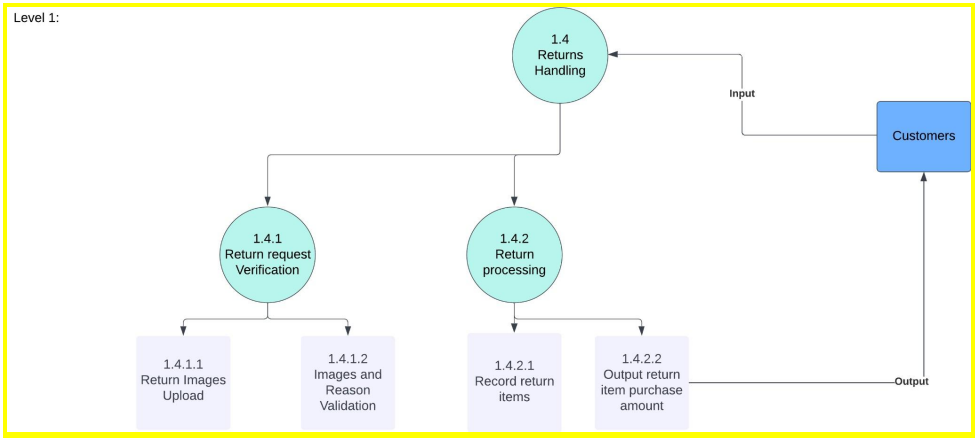
A. Bird's Eye View of Data Flow



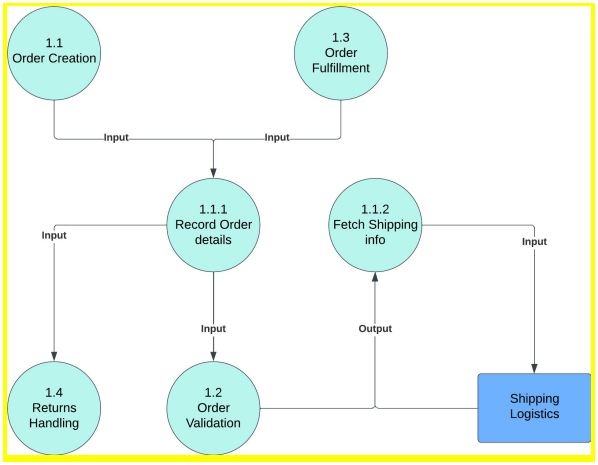
B. Order Processing System Detailed Data Flow Diagram



C. Customers Data Flow Diagram



D. Shipping Logistics Data Flow Diagram



3. Database Schema (includes all tables, fields)

GBC_Superstores DB

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length
customer	InnoDB	10	Dynamic	793	144	112.0 KiB	0.0 bytes
order	InnoDB	10	Dynamic	4946	96	464.0 KiB	0.0 bytes
order_item	InnoDB	10	Dynamic	9955	159	1.5 MiB	0.0 bytes
product	InnoDB	10	Dynamic	98	167	16.0 KiB	0.0 bytes
region	InnoDB	10	Dynamic	4	4096	16.0 KiB	0.0 bytes
returns	InnoDB	10	Dynamic	800	81	64.0 KiB	0.0 bytes

Customer Table :

Column	Type	Default Value	Nullable	Character Set	Collation
City	varchar(45)		YES	utf8mb4	utf8mb4_090...
Country	varchar(45)		YES	utf8mb4	utf8mb4_090...
Customer Name	varchar(45)		YES	utf8mb4	utf8mb4_090...
customerId	varchar(45)		NO	utf8mb4	utf8mb4_090...
Postal Code	int		YES		
Segment	varchar(45)		YES	utf8mb4	utf8mb4_090...
State	varchar(45)		YES	utf8mb4	utf8mb4_090...

Order Table:

Column	Type	Default Value	Nullable	Character Set	Collation
CustomerID	varchar(45)		YES	utf8mb4	utf8mb4_090...
Order Date	datetime		YES		
OrderID	varchar(255)		NO	utf8mb4	utf8mb4_090...
Region	int		YES		
Ship Date	datetime		YES		
Ship Mode	text		YES	utf8mb4	utf8mb4_090...

Order Item Table

Column	Type	Default Value	Nullable	Character Set	Collation
Cost of Goods Sold	double		YES		
Cost of Sales	double		YES		
Discount	double		YES		
order_itemId	varchar(64)		NO	utf8mb4	utf8mb4_090...
OrderID	varchar(64)		YES	utf8mb4	utf8mb4_090...
Other Expenses	double		YES		
ProductID	varchar(64)		YES	utf8mb4	utf8mb4_090...
Profit	double		YES		
Profit Margin	double		YES		
Quantity	int		YES		
Sales	double		YES		

Product Table:

Column	Type	Default Value	Nullable	Character Set	Collation
Category	text		YES	utf8mb4	utf8mb4_090...
Product Name	text		YES	utf8mb4	utf8mb4_090...
ProductID	varchar(64)		NO	utf8mb4	utf8mb4_090...
Sub-Category	text		YES	utf8mb4	utf8mb4_090...

Region Table:

Column	Type	Default Value	Nullable	Character Set	Collation
Region	text		YES	utf8mb4	utf8mb4_090...
Regional Manager	text		YES	utf8mb4	utf8mb4_090...
RegionID	int		NO		

Returns Table:

Column	Type	Default Value	Nullable	Character Set	Collation
OrderID	varchar(64)		YES	utf8mb4	utf8mb4_090...
ReturnsID	varchar(64)		NO	utf8mb4	utf8mb4_090...

Database Schema and Table Schema

```
CREATE TABLE `customer` (
  `customerId` varchar(45) NOT NULL,
  `Customer Name` varchar(45) DEFAULT NULL,
  `Segment` varchar(45) DEFAULT NULL,
  `Country` varchar(45) DEFAULT NULL,
  `City` varchar(45) DEFAULT NULL,
  `State` varchar(45) DEFAULT NULL,
  `Postal Code` int DEFAULT NULL,
  PRIMARY KEY (`customerId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `order` (
  `OrderID` varchar(255) NOT NULL,
  `Order Date` datetime DEFAULT NULL,
  `Ship Date` datetime DEFAULT NULL,
  `Ship Mode` text,
  `Region` int DEFAULT NULL,
  `CustomerID` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`OrderID`),
  KEY `CustomerID` (`CustomerID`),
  KEY `Region` (`Region`),
  CONSTRAINT `order_ibfk_1` FOREIGN KEY (`CustomerID`) REFERENCES
`customer` (`customerId`),
  CONSTRAINT `order_ibfk_2` FOREIGN KEY (`Region`) REFERENCES `region`
(`RegionID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `order_item` (
  `order_itemId` varchar(64) NOT NULL,
  `OrderID` varchar(64) DEFAULT NULL,
  `ProductID` varchar(64) DEFAULT NULL,
  `Sales` double DEFAULT NULL,
  `Quantity` int DEFAULT NULL,
  `Discount` double DEFAULT NULL,
  `Profit` double DEFAULT NULL,
  `Profit Margin` double DEFAULT NULL,
  `Cost of Goods Sold` double DEFAULT NULL,
  `Cost of Sales` double DEFAULT NULL,
```

Name for current table column and reference table column can be same.

```
  `Other Expenses` double DEFAULT NULL,
  PRIMARY KEY (`order_itemId`),
  KEY `OrderID` (`OrderID`),
  KEY `ProductID` (`ProductID`),
  CONSTRAINT `order_item_ibfk_1` FOREIGN KEY (`OrderID`) REFERENCES
`order` (`OrderID`),
  CONSTRAINT `order_item_ibfk_2` FOREIGN KEY (`ProductID`) REFERENCES
`product` (`ProductID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `product` (
  `ProductID` varchar(64) NOT NULL,
  `Category` text,
  `Sub-Category` text,
  `Product Name` text,
  PRIMARY KEY (`ProductID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `region` (
  `RegionID` int NOT NULL,
  `Region` text,
  `Regional Manager` text,
  PRIMARY KEY (`RegionID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `returns` (
  `ReturnsID` varchar(64) NOT NULL,
  `OrderID` varchar(64) DEFAULT NULL,
  PRIMARY KEY (`ReturnsID`),
  KEY `OrderID` (`OrderID`),
  CONSTRAINT `returns_ibfk_1` FOREIGN KEY (`OrderID`) REFERENCES `order`
(`OrderID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

4. **Documentation of any data source changes**

   We added some additional columns on the source data based on certain assumptions. Assumptions and calculation of the new columns are listed below:

   a. We assume the amount in the “Sales” column is the combination of cost of goods sold, cost of sales, other expenses, profit, and discount. Thus, we are defining sale column as,

      a. Cost of Goods Sold (COGS) + Cost of Sales + Profit + Other Expenses - Discounts = Sales Revenue

b. In addition to that, we assume for every sale, after excluding profit and discount, COGS will contribute to 2x of the sale amount and Cost of sales and other expense will contribute to x/2 of the sale amount. For example,
    a. If sale=100%, discount is 1% and profit after discount is 15%, then remaining 85% of value will be splitted to COGS, Cost of sales, and for the other expenses.
    b. Where COGS will contribute to (85/3)*2 = 56.67% of sales value, Cost of sales will contribute to (85/3)*1 = 28.34% of sales value, and other expense will contribute to (85/3)*1 = 28.34% of sales value.
c. By defining above calculation, we are adding three new columns in the source data that is listed below:
    a. Cost of Goods Sold(COGS)
    b. Cost of sales
    c. Other Expenses
d. We found that data is about order item instead of order itself. Because, we could find multiple same order ids. It is because of the fact that one order id could have multiple order item id. So, We have created the *Order Item Id* by concatenating row id and order id.
e. We also created the *Return ID* by concatenating “R” with order id. Also, we assume returns is for order id instead of order item id.
f. We have also added Region ID which could serve as the primary key on the Region table.

5. Documentation explaining the ETL process

a. We have to split the one source table into multiple entities, as splitted on the logical ERD.
b. We are using Python and Pandas library for ETL process and exporting all the processed entities data into CSV which could be imported in the MYSQL tables which were created earlier.
c. Building product entity by grouping the data with the Product ID column in the source data.

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

```
product = order[['Product ID', 'Category', 'Sub-Category', 'Product Name']].groupby("Product ID", as_index=False).first
```

```
product.head()
```

	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

d. Building Region entity by using data from the “People” sheet in the source data. We additionally created Region ID using row indexes that could be served as primary key for for region table.

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

```
region = pd.read_excel("/Users/vignesh/Documents/george brown pgdm /Foundation of data management/Lab Exercises/Sample
```

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

```
region.head()
```

	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

e. It extracts specific columns from an "order" table, including sales, quantity, discount, profit, and other relevant information for order items. It generates a unique "Order Item ID" for each order item by concatenating the "Order ID" and "Row ID" as a string. This process prepares the data for representing the one-to-many relationship, where one order can have multiple order items, each uniquely identified by "Order Item ID."

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

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

```
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', 'Profit
       'Cost of Goods Sold', 'Cost of Sales', 'Other Expenses']]
```

```
order_item.head()
```

	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

f. Building customer entity by grouping the data with the Customer ID column in the source data.

One to many relationship: One customer can have multiple orders

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

```
customer.head()
```

	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

6. SQL scripts, CSV files, and/or Python code used for ETL

A. SQL Scripts

Github Link:
https://github.com/RoanKathrina/Foundations-of-Data-Management/tree/main/Assignment%202/SQL%20Queries

B. CSV Files

a. Order Table

Github Link:
https://raw.githubusercontent.com/vignesh865/foundation-of-data-management/main/Order%20table1.csv

b. Product Table

Github Link:
https://raw.githubusercontent.com/vignesh865/foundation-of-data-management/main/Product%20table1.csv

c. Region Table

Github Link:
https://raw.githubusercontent.com/vignesh865/foundation-of-data-management/main/Region%20table1.csv

d. Order Item Table

Github Link:
https://raw.githubusercontent.com/vignesh865/foundation-of-data-management/main/Order%20Item%20table1.csv

e. Customer Table

Github Link:
https://raw.githubusercontent.com/vignesh865/foundation-of-data-management/main/Customer%20table5.csv

f. Returns Table

Github Link:
https://raw.githubusercontent.com/vignesh865/foundation-of-data-management/main/Returns%20table.csv

C. Python Source Code for ETL Processing

Github Link:

https://github.com/RoanKathrina/Foundations-of-Data-Management/tree/main/Assignment%202/Python%20ETL%20Source%20Code

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…

```
CREATE TABLE `customer` (
  `customerId` varchar(45) NOT NULL,
  `Customer Name` varchar(45) DEFAULT NULL,
  `Segment` varchar(45) DEFAULT NULL,
  `Country` varchar(45) DEFAULT NULL,
  `City` varchar(45) DEFAULT NULL,
  `State` varchar(45) DEFAULT NULL,
  `Postal Code` int DEFAULT NULL,
  PRIMARY KEY (`customerId`)
) ;


CREATE TABLE `order` (
  `OrderID` varchar(255) NOT NULL,
  `Order Date` datetime DEFAULT NULL,
  `Ship Date` datetime DEFAULT NULL,
  `Ship Mode` text,
  `Region` int DEFAULT NULL,
  `CustomerID` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`OrderID`),
  KEY `CustomerID` (`CustomerID`),
  KEY `Region` (`Region`),
  CONSTRAINT `order_ibfk_1` FOREIGN KEY (`CustomerID`) REFERENCES
`customer` (`customerId`),
  CONSTRAINT `order_ibfk_2` FOREIGN KEY (`Region`) REFERENCES
`region` (`RegionID`)
) ;

CREATE TABLE `order_item` (
  `order_itemId` varchar(64) NOT NULL,
  `OrderID` varchar(64) DEFAULT NULL,
  `ProductID` varchar(64) DEFAULT NULL,
  `Sales` double DEFAULT NULL,
  `Quantity` int DEFAULT NULL,
  `Discount` double DEFAULT NULL,
  `Profit` double DEFAULT NULL,
  `Profit Margin` double DEFAULT NULL,
  `Cost of Goods Sold` double DEFAULT NULL,
  `Cost of Sales` double DEFAULT NULL,
  `Other Expenses` double DEFAULT NULL,
```

```
  PRIMARY KEY (`order_itemId`),
  KEY `OrderID` (`OrderID`),
  KEY `ProductID` (`ProductID`),
  CONSTRAINT `order_item_ibfk_1` FOREIGN KEY (`OrderID`)
REFERENCES `order` (`OrderID`),
  CONSTRAINT `order_item_ibfk_2` FOREIGN KEY (`ProductID`)
REFERENCES `product` (`ProductID`)
) ;

CREATE TABLE `product` (
  `ProductID` varchar(64) NOT NULL,
  `Category` text,
  `Sub-Category` text,
  `Product Name` text,
  PRIMARY KEY (`ProductID`)
) ;

CREATE TABLE `region` (
  `RegionID` int NOT NULL,
  `Region` text,
  `Regional Manager` text,
  PRIMARY KEY (`RegionID`)
) ;

CREATE TABLE `returns` (
  `ReturnsID` varchar(64) NOT NULL,
  `OrderID` varchar(64) DEFAULT NULL,
  PRIMARY KEY (`ReturnsID`),
  KEY `OrderID` (`OrderID`),
  CONSTRAINT `returns_ibfk_1` FOREIGN KEY (`OrderID`) REFERENCES
`order` (`OrderID`)
) ;
```