In [1]:
```
!pip install kaggle
import kaggle
```

```
Requirement already satisfied: kaggle in c:\users\a\anaconda3\lib\site-packag
es (1.6.14)
Requirement already satisfied: python-dateutil in c:\users\a\anaconda3\lib\si
te-packages (from kaggle) (2.8.2)
Requirement already satisfied: python-slugify in c:\users\a\anaconda3\lib\sit
e-packages (from kaggle) (5.0.2)
Requirement already satisfied: certifi>=2023.7.22 in c:\users\a\anaconda3\lib
\site-packages (from kaggle) (2024.6.2)
Requirement already satisfied: urllib3 in c:\users\a\anaconda3\lib\site-packa
ges (from kaggle) (1.26.9)
Requirement already satisfied: requests in c:\users\a\anaconda3\lib\site-pack
ages (from kaggle) (2.27.1)
Requirement already satisfied: bleach in c:\users\a\anaconda3\lib\site-packag
es (from kaggle) (4.1.0)
Requirement already satisfied: six>=1.10 in c:\users\a\anaconda3\lib\site-pac
kages (from kaggle) (1.16.0)
Requirement already satisfied: tqdm in c:\users\a\anaconda3\lib\site-packages
(from kaggle) (4.64.0)
Requirement already satisfied: webencodings in c:\users\a\anaconda3\lib\site-
packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: packaging in c:\users\a\anaconda3\lib\site-pac
kages (from bleach->kaggle) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\a\anacond
a3\lib\site-packages (from packaging->bleach->kaggle) (3.0.4)
Requirement already satisfied: text-unidecode>=1.3 in c:\users\a\anaconda3\li
b\site-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<4,>=2.5 in c:\users\a\anaconda3\lib\site-
packages (from requests->kaggle) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\a\anacon
da3\lib\site-packages (from requests->kaggle) (2.0.4)
Requirement already satisfied: colorama in c:\users\a\anaconda3\lib\site-pack
ages (from tqdm->kaggle) (0.4.4)
```

In [2]:
```
#downloading data set
!kaggle datasets download ankitbansal06/retail-orders -f orders.csv
```

```
Dataset URL: https://www.kaggle.com/datasets/ankitbansal06/retail-orders (htt
ps://www.kaggle.com/datasets/ankitbansal06/retail-orders)
License(s): CC0-1.0
orders.csv.zip: Skipping, found more recently modified local copy (use --forc
e to force download)
```

In [3]:
```
#extracting zip file
import zipfile
zip_ref = zipfile.ZipFile('orders.csv.zip')
zip_ref.extractall()
zip_ref.close()
```

In [4]:
```
#READ DATA
import pandas as pd
df = pd.read_csv('orders.csv.zip',na_values= ['Not Available', 'unknown'])
df.head(20)
```

Out[4]:

	Order Id	Order Date	Ship Mode	Segment	Country	City	State	Postal Code	Region	Ca
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Fu
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Fu
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Su
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Fu
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Su
5	6	2022-03-13	NaN	Consumer	United States	Los Angeles	California	90032	West	Fu
6	7	2022-12-28	Standard Class	Consumer	United States	Los Angeles	California	90032	West	Su

In [5]:
```
df['Ship Mode'].unique()
```

Out[5]:
```
array(['Second Class', 'Standard Class', nan, 'First Class', 'Same Day'],
      dtype=object)
```

In [6]:
```
#lowercase + replacing spaces
df.columns= df.columns.str.lower()
df.columns= df.columns.str.replace(' ','_')
df.columns
df.head(5)
```

Out[6]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	regi
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	Sou
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	Sou
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	We
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	Sou
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	Sou

In [7]:
```
#new columns = discount ,  sale price , profit

df['discount']= df['list_price']*df['discount_percent']*.01
df["sale_price"]= df['list_price']-df['discount']

df["profit"]= df["sale_price"]- df['cost_price']
df.head(5)
```

Out[7]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	regi
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	Sou
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	Sou
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	We
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	Sou
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	Sou

In [8]:
```
#checking the data type
print(df.dtypes)
```

```
order_id              int64
order_date           object
ship_mode            object
segment              object
country              object
city                 object
state                object
postal_code           int64
region               object
category             object
sub_category         object
product_id           object
cost_price            int64
list_price            int64
quantity              int64
discount_percent      int64
discount            float64
sale_price          float64
profit              float64
dtype: object
```

In [9]:
```
#converting order date to datetime data type rather than object
df["order_date"]= pd.to_datetime(df["order_date"],format= "%Y-%m-%d")
print(df.dtypes)
```

```
order_id                     int64
order_date          datetime64[ns]
ship_mode                   object
segment                     object
country                     object
city                        object
state                       object
postal_code                  int64
region                      object
category                    object
sub_category                object
product_id                  object
cost_price                   int64
list_price                   int64
quantity                     int64
discount_percent             int64
discount                   float64
sale_price                 float64
profit                     float64
dtype: object
```

In [10]:
```
#dropping unwanted columns for the analysis
df.drop(columns=["list_price", "discount_percent", 'cost_price' ],inplace = Tr
df
```

Out[10]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311
...	...	...	...	...	...	...	...	...
9989	9990	2023-02-18	Second Class	Consumer	United States	Miami	Florida	33180
9990	9991	2023-03-17	Standard Class	Consumer	United States	Costa Mesa	California	92627
9991	9992	2022-08-07	Standard Class	Consumer	United States	Costa Mesa	California	92627
9992	9993	2022-11-19	Standard Class	Consumer	United States	Costa Mesa	California	92627
9993	9994	2022-07-17	Second Class	Consumer	United States	Westminster	California	92683

9994 rows × 16 columns

In [11]:
```
! pip install pyodbc
import pyodbc

# Get the list of ODBC drivers installed on the system
drivers = pyodbc.drivers()

print("Available ODBC Drivers:")
for driver in drivers:
    print(driver)
```

```
Requirement already satisfied: pyodbc in c:\users\a\anaconda3\lib\site-packag
es (4.0.32)
Available ODBC Drivers:
SQL Server
Microsoft Access Driver (*.mdb, *.accdb)
Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
Microsoft Access Text Driver (*.txt, *.csv)
MySQL ODBC 8.0 ANSI Driver
MySQL ODBC 8.0 Unicode Driver
SQL Server Native Client RDA 11.0
ODBC Driver 17 for SQL Server
```

In [12]:
```
import sqlalchemy as sal
from sqlalchemy import create_engine

# Define your connection parameters
server = 'DESKTOP-RP1V4PE\\SQLEXPRESS01'
database = 'comrade'
driver = 'ODBC Driver 17 for SQL Server'

# Create the connection string for Windows Authentication
connection_string = f"mssql+pyodbc://@{server}/{database}?driver={driver}&trus

# Create an engine
engine = create_engine(connection_string)

# Test the connection
try:
    with engine.connect() as connection:
        print("Connection successful!")
except Exception as e:
    print(f"Connection failed: {e}")
```

```
Connection successful!
```

In [14]:
```
conn= engine.connect()
df.to_sql('df_orders',con=conn,index= False,if_exists = 'append')
```

Out[14]:
```
-1
```