## 1. Import required libraries

In [1]:
```
import kaggle
import pandas as pd
import numpy as np
```

executed in 1.18s, finished 20:00:09 2024-05-31

## 2. Download dataset using kaggle

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

executed in 3.97s, finished 20:00:13 2024-05-31

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

## 3. Extract file from zip file

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

executed in 16ms, finished 20:00:13 2024-05-31

## 4. Read data from the file and handle null values

In [4]:
```
#reading data
df = pd.read_csv('orders.csv')
df.head(20)
```

executed in 93ms, finished 20:00:13 2024-05-31

Out[4]:

| | Order Id | Order Date | Ship Mode | Segment | Country | City | State | Postal Code | Region |
|---|---|---|---|---|---|---|---|---|---|
| **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 |
| **5** | 6 | 2022-03-13 | Not Available | Consumer | United States | Los Angeles | California | 90032 | We |
| **6** | 7 | 2022-12-28 | Standard Class | Consumer | United States | Los Angeles | California | 90032 | We |
| **7** | 8 | 2022-01-25 | Standard Class | Consumer | United States | Los Angeles | California | 90032 | We |
| **8** | 9 | 2023-03-23 | Not Available | Consumer | United States | Los Angeles | California | 90032 | We |
| **9** | 10 | 2023-05-16 | Standard Class | Consumer | United States | Los Angeles | California | 90032 | We |
| **10** | 11 | 2023-03-31 | Not Available | Consumer | United States | Los Angeles | California | 90032 | We |
| **11** | 12 | 2023-12-25 | Not Available | Consumer | United States | Los Angeles | California | 90032 | We |
| **12** | 13 | 2022-02-11 | Standard Class | Consumer | United States | Concord | North Carolina | 28027 | Sou |
| **13** | 14 | 2023-07-18 | Standard Class | Consumer | United States | Seattle | Washington | 98103 | We |
| **14** | 15 | 2023-11-09 | unknown | Home Office | United States | Fort Worth | Texas | 76106 | Centr |
| **15** | 16 | 2022-06-18 | Standard Class | Home Office | United States | Fort Worth | Texas | 76106 | Centr |
| **16** | 17 | 2022-02-04 | Standard Class | Consumer | United States | Madison | Wisconsin | 53711 | Centr |
| **17** | 18 | 2023-08-04 | Second Class | Consumer | United States | West Jordan | Utah | 84084 | We |
| **18** | 19 | 2022-01-23 | Second Class | Consumer | United States | San Francisco | California | 94109 | We |
| **19** | 20 | 2022-01-11 | Second Class | Consumer | United States | San Francisco | California | 94109 | We |

In [5]:
```
#handlling null values
df = pd.read_csv('orders.csv', na_values=['Not Available', 'unknown'])
df['Ship Mode'].unique()
```
executed in 57ms, finished 20:00:14 2024-05-31

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

## 5. Rename columns names.. (make them lower case and replace space with underscore)

In [6]:
```
#name of the columns
df.columns
```
executed in 9ms, finished 20:00:14 2024-05-31

Out[6]:
```
Index(['Order Id', 'Order Date', 'Ship Mode', 'Segment', 'Country', 'City',
       'State', 'Postal Code', 'Region', 'Category', 'Sub Category',
       'Product Id', 'cost price', 'List Price', 'Quantity',
       'Discount Percent'],
      dtype='object')
```

In [7]:
```
#making columns to lower case
df.columns= df.columns.str.lower()
df.head(2)
```
executed in 24ms, finished 20:00:14 2024-05-31

Out[7]:

| | 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 | |
| **1** | 2 | 2023-08-15 | Second Class | Consumer | United States | Henderson | Kentucky | 42420 | South | |

In [8]:
```
#replacing space with uderscore
df.columns= df.columns.str.replace(' ', '_')
df.head(2)
```
executed in 29ms, finished 20:00:14 2024-05-31

Out[8]:

| | order_id | order_date | ship_mode | segment | country | city | state | postal_co |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2023-03-01 | Second Class | Consumer | United States | Henderson | Kentucky | |
| **1** | 2 | 2023-08-15 | Second Class | Consumer | United States | Henderson | Kentucky | |

## 6. Drive a new columns discount, sale price and profit

In [9]:
```
#creating a discount column
df['discount']= df['list_price']*df['discount_percent']*.01
```
executed in 17ms, finished 20:00:14 2024-05-31

In [10]:
```
#creating a sale price columns
df['sale_price']= df['list_price']-df['discount']
```
executed in 10ms, finished 20:00:14 2024-05-31

In [11]:
```
#creating a profit column
df['profit']= df['sale_price']-df['cost_price']
```
executed in 12ms, finished 20:00:14 2024-05-31

In [12]:
```
#reading the file
df.head(5)
```
executed in 35ms, finished 20:00:14 2024-05-31

Out[12]:

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

## 7. Convert order date from object data type to datetime

In [13]:
```
#converting data type
df['order_date'] = pd.to_datetime(df['order_date'], format="%Y-%m-%d")
```
executed in 25ms, finished 20:00:14 2024-05-31

In [14]:
```
#checking the data types
df.dtypes
```
executed in 17ms, finished 20:00:14 2024-05-31

Out[14]:
```
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
```

## 8. Drop cost price, list price and discount percent columns

In [15]:
```
df.drop(columns=['cost_price', 'list_price', 'discount_percent'], inplace=True
```
executed in 12ms, finished 20:00:14 2024-05-31

In [16]:
```
#checking the columns after droping
df.columns
```

executed in 23ms, finished 20:00:14 2024-05-31

Out[16]:
```
Index(['order_id', 'order_date', 'ship_mode', 'segment', 'country', 'city',
       'state', 'postal_code', 'region', 'category', 'sub_category',
       'product_id', 'quantity', 'discount', 'sale_price', 'profit'],
      dtype='object')
```

## 9. Load the data into sql server

In [18]:
```
from pandas.io import sql
from sqlalchemy import create_engine

engine = create_engine("mysql+pymysql://{user}:{pw}@localhost:3306/{db}"
                       .format(user="root",
                               pw='Suhaib200',
                               db="ecommerce"))

df.to_sql(con=engine, name='ecommerce', if_exists='append', index = False)
```

executed in 536ms, finished 21:12:44 2024-05-31

Out[18]:
```
9994
```

```
-- Creating an ecommerce table
create table ecommerce (
                order_id int primary key,
        order_date date,
        ship_mode varchar(20),
        segment varchar(20),
        country varchar(20),
        city varchar(20),
        state varchar(20),
        postal_code varchar(20),
        region varchar(20),
        category varchar(20),
        sub_category varchar(20),
        product_id varchar(50),
        quantity int,
        discount decimal(7,2),
        sale_price decimal(7,2),
        profit decimal(7,2));



-- find top 10 highest reveue generating products
SELECT product_id, SUM(sale_price) AS sales
FROM ecommerce
GROUP BY product_id
ORDER BY sales DESC
LIMIT 10;



-- find top 5 highest selling products in each region
SELECT region, product_id, total_sales
FROM (
    SELECT region, product_id, SUM(sale_price) as total_sales,
    RANK() OVER (PARTITION BY region ORDER BY SUM(sale_price) DESC) as
sales_rank
    FROM ecommerce
    GROUP BY region, product_id
) as sales_ranks
WHERE sales_rank <= 5;



-- find month over month growth comparison for 2022 and 2023 sales eg :
jan 2022 vs jan 2023
WITH cte AS (
SELECT year(order_date) AS order_year, MONTH(order_date) AS order_month,
SUM(sale_price) AS sales
FROM ecommerce
GROUP BY YEAR(order_date), MONTH(order_date)
)
SELECT order_month
, SUM(CASE WHEN order_year=2022 THEN sales ELSE 0 END) AS sales_2022
, SUM(CASE WHEN order_year=2023 THEN sales ELSE 0 END) AS sales_2023
```

```
FROM cte
GROUP BY order_month
ORDER BY order_month;



-- for each category which month had highest sales
 WITH cte AS (
SELECT category, FORMAT(order_date,'yyyyMM') AS order_year_month
, SUM(sale_price) AS sales
FROM ecommerce
GROUP BY category, FORMAT(order_date,'yyyyMM')
)
SELECT * FROM(
SELECT *,
ROW_NUMBER() OVER(PARTITION BY category ORDER BY sales DESC) AS rn
FROM cte
) a
WHERE rn=1;



-- which sub category had highest growth by profit in 2023 compare to 2022
WITH cte AS (
    SELECT
        sub_category,
        YEAR(order_date) AS order_year,
        SUM(sale_price) AS sales
    FROM
        ecommerce
    GROUP BY
        sub_category, YEAR(order_date)
),
cte2 AS (
    SELECT
        sub_category,
        SUM(CASE WHEN order_year = 2022 THEN sales ELSE 0 END) AS
sales_2022,
        SUM(CASE WHEN order_year = 2023 THEN sales ELSE 0 END) AS
sales_2023
    FROM
        cte
    GROUP BY
        sub_category
)
SELECT
    *,
    (sales_2023 - sales_2022) AS sales_difference
FROM
    cte2
ORDER BY
    sales_difference DESC
LIMIT 1;
```