Retail Orders Analysis

Import Libraries

1. kaggle: This library is used to access and download datasets directly from Kaggle. It's particularly helpful for retrieving competition or public datasets in Jupyter notebooks.

2. zipfile: This module is used to work with .zip files. It helps extract compressed files downloaded from Kaggle or elsewhere, making them ready for use.

3. pandas: A powerful library for data manipulation and analysis. It is commonly used to read, clean, and process datasets (e.g., CSV files) into DataFrames, which are easy to analyze.

4. sqlalchemy: This library is used to interact with databases. It helps create a database engine, allowing you to store, query, or retrieve data from databases in a Pythonic way.

In [1]:
```
#import libraries
import kaggle
import zipfile
import pandas as pd
from sqlalchemy import create_engine
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /Users/sauravghag/.kaggle/kaggle.json'
```

Downloading Kaggle dataset

Below command downloads the `orders.csv` file from the Kaggle dataset `retail-orders` into your current working directory.

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

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /Users/sauravghag/.kaggle/kaggle.json'
Dataset URL: 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)
```

Extraction of the downloaded zip file

This code is handling the extraction of a .zip file. It extracts the orders.csv file (or other files if present) from the compressed orders.csv.zip file into your working directory.

- zipfile.ZipFile('orders.csv.zip'): Opens the orders.csv.zip file for reading. This file contains the compressed dataset.
- zip_file.extractall(): Extracts all the contents of the .zip file into the current directory (or a specified directory if given).
- zip_file.close(): Closes the .zip file to free up system resources.

In [3]:
```
# Extract the downloaded zip file
zip_file = zipfile.ZipFile('orders.csv.zip')
zip_file.extractall() # Extract file to directory
zip_file.close() # Close File
```

Loading and Cleaning of Dataset

We will now perform the loading of the downloaded dataset and cleaning of the raw data. This script processes the dataset by cleaning, renaming columns, calculating new fields, converting data types, and removing unnecessary columns, making it ready for analysis.

1. Load the dataset:

- `data = pd.read_csv('orders.csv')` Reads the `orders.csv` file into a Pandas DataFrame.

1. Check unique values in Ship Mode column:

- `data['Ship Mode'].unique()` Lists all unique values in the column named `Ship Mode`.

1. Replace specific values with `NaN`:

- `data = pd.read_csv('orders.csv', na_values=['Not Available', 'unknown'])` While loading the data, values like Not Available and unknown are replaced with NaN (missing values).

1. Rename columns to a consistent format: `data.columns = [column.lower().replace(' ', '_') for column in data]` Converts column names to lowercase and replaces spaces with underscores for easier handling.

2. Derive new columns:

- `discount`: The discount amount is calculated as `(list_price * discount_percent) / 100`.
- `sell_price`: The selling price is derived as `list_price - discount`.
- `profit`: The profit is calculated as `sell_price - cost_price`.

1. View data and types:

- `data.head()` displays the first 5 rows of the dataset.
- `data.dtypes` shows the data types of each column.

1. Convert `order_date` to a `datetime` type:

- `data['order_date'] = pd.to_datetime(data['order_date'], format="%Y-%m-%d")` Converts the `order_date` column to a `datetime` object for easier date operations.

1. Drop unnecessary columns:

- `data.drop(columns=['cost_price', 'list_price', 'discount_percent'], inplace=True)` Removes the specified columns (`cost_price, list_price, discount_percent`) from the DataFrame to clean up the data.

In [4]:
```
# Load dataset
data = pd.read_csv('orders.csv')
data
```

Out[4]:

	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
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2	2
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3	3
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	2	5
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	5	2
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	OFF-ST-10000760	20	20	2	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9989	9990	2023-02-18	Second Class	Consumer	United States	Miami	Florida	33180	South	Furniture	Furnishings	FUR-FU-10001889	30	30	3	4
9990	9991	2023-03-17	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Furniture	Furnishings	FUR-FU-10000747	70	90	2	4
9991	9992	2022-08-07	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Technology	Phones	TEC-PH-10003645	220	260	2	2
9992	9993	2022-11-19	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Office Supplies	Paper	OFF-PA-10004041	30	30	4	3
9993	9994	2022-07-17	Second Class	Consumer	United States	Westminster	California	92683	West	Office Supplies	Appliances	OFF-AP-10002684	210	240	2	3

9994 rows × 16 columns

In [5]:
```
# Checking unique values in column Shipmode
data['Ship Mode'].unique()
```

Out[5]:
```
array(['Second Class', 'Standard Class', 'Not Available', 'unknown',
       'First Class', nan, 'Same Day'], dtype=object)
```

In [6]:
```
# Replacing 'Not Available', 'unknown' as nan
data = pd.read_csv('orders.csv', na_values=['Not Available', 'unknown'])
print(data['Ship Mode'].unique())
data
```

```
['Second Class' 'Standard Class' nan 'First Class' 'Same Day']
```

Out[6]:

	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
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2	2
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3	3
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	2	5
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	5	2
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	OFF-ST-10000760	20	20	2	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9989	9990	2023-02-18	Second Class	Consumer	United States	Miami	Florida	33180	South	Furniture	Furnishings	FUR-FU-10001889	30	30	3	4
9990	9991	2023-03-17	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Furniture	Furnishings	FUR-FU-10000747	70	90	2	4
9991	9992	2022-08-07	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Technology	Phones	TEC-PH-10003645	220	260	2	2
9992	9993	2022-11-19	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Office Supplies	Paper	OFF-PA-10004041	30	30	4	3
9993	9994	2022-07-17	Second Class	Consumer	United States	Westminster	California	92683	West	Office Supplies	Appliances	OFF-AP-10002684	210	240	2	3

9994 rows × 16 columns

In [7]:
```
# We need to rename column_names to lower case and replace space with '_'
#df.rename(columns={'Order Id':'order_id', 'City':'city'})
#df.columns=df.columns.str.lower()
#df.columns=df.columns.str.replace(' ','_')

# Doing the above in single line
data.columns = [column.lower().replace(' ', '_') for column in data]
data.head()
```

Out[7]:

	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
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2	2
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3	3
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	2	5
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	5	2
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	OFF-ST-10000760	20	20	2	5

In [8]:
```
# Derive new columns, discount, sell_price and profit
data['discount'] = (data['list_price']*data['discount_percent'])/100
data['sell_price'] = data['list_price'] - data['discount']
data['profit'] = data['sell_price'] - data['cost_price']
```

In [9]:
```
data.head()
```

Out[9]:

	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	discount	sell_price	profit
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2	2	5.2	254.8	14.8
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3	3	21.9	708.1	108.1
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	2	5	0.5	9.5	-0.5
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	5	2	19.2	940.8	160.8
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	OFF-ST-10000760	20	20	2	5	1.0	19.0	-1.0

In [10]:
```
data.dtypes
```

Out[10]:
```
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
sell_price          float64
profit              float64
dtype: object
```

In [11]:
```
# Convert order_data data type from object to datetime
data['order_date'] = pd.to_datetime(data['order_date'], format="%Y-%m-%d")
```

In [12]:
```
data.dtypes
```

Out[12]:
```
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
sell_price                 float64
profit                     float64
dtype: object
```

In [13]:
```
data.head()
```

Out[13]:

	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	discount	sell_price	profit
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2	2	5.2	254.8	14.8
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3	3	21.9	708.1	108.1
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	2	5	0.5	9.5	-0.5
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	5	2	19.2	940.8	160.8
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	OFF-ST-10000760	20	20	2	5	1.0	19.0	-1.0

In [14]:
```
# Drop cost_price, list_price and discount_percent columns in the data
data.drop(columns=['cost_price', 'list_price', 'discount_percent'], inplace=True)
data.head()
```

Out[14]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_category	product_id	quantity	discount	sell_price	profit
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	2	5.2	254.8	14.8
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	3	21.9	708.1	108.1
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	2	0.5	9.5	-0.5
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	5	19.2	940.8	160.8
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	OFF-ST-10000760	2	1.0	19.0	-1.0

Connecting to SQL Server and Loading Data into the Database

This code establishes a connection to a MySQL database and uploads the `data` DataFrame into a table. By storing the processed dataset in the database, it facilitates efficient querying, analysis, and reporting through SQL.

1. Create the database engine:
   ```
   engine = create_engine('mysql+pymysql://root:root1234@localhost:3306/retail_data_analysis')
   ```

- Uses SQLAlchemy to connect to a MySQL database.
- Connection string format: `dialect+driver://username:password@host:port/database`.
- Example: Connects as `root` with password `root1234` to the database `retail_data_analysis` on `localhost` at port `3306`.

1. Establish the connection:
   ```
   conn = engine.connect()
   ```

- Opens a connection to the database.
- Ensures the script can interact with the database.

1. Upload the data to SQL:
   ```
   data.to_sql('retail_orders', con=conn, index=False, if_exists='append')
   ```

- Saves the `data` DataFrame to a table named `retail_orders` in the database.
- `index=False`: Excludes the DataFrame index column when writing to the table.
- `if_exists='append'`: Appends the data to the table if it already exists (instead of replacing it).

In [15]:
```
# Connect to SQL Server and Database
engine = create_engine('mysql+pymysql://root:root1234@localhost:3306/retail_data_analysis')
conn = engine.connect()
print("Connected successfully!")
```

```
Connected successfully!
```

In [16]:
```
# Load the data into SQL Server using replace option
data.to_sql('retail_orders', con=conn, index=False, if_exists='append')
```

Out[16]:
```
9994
```

Cleaned Data to New Dataset

This line of code exports the `data` DataFrame to a CSV file named `retail_order.csv` in the current working directory, making it available for storage, sharing, or further use.

- `to_csv`: A Pandas method to export a DataFrame to a CSV file.
- `"retail_order.csv"`: The name of the output file.
- `index=False`: Excludes the DataFrame's index column from the saved file.

In [17]:
```
# This command was used to make a new dataset of cleaned data
data.to_csv("retail_order.csv", index=False)
```