```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt  # Data visualization library
import seaborn as sns  # Data visualization library for creating informative and attractive statistical graphics

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/supermarket-sales-data/annex1.csv
/kaggle/input/supermarket-sales-data/annex3.csv
/kaggle/input/supermarket-sales-data/annex2.csv
/kaggle/input/supermarket-sales-data/annex4.csv
```

Project Summary

Welcome to the Sectional Project derived from the main KaggleX Mentorship Final Project Report, titled

In this comprehensive project, I have deconstructed the primary project into six distinct EDA sections, each finely tuned to focus on a specific aspect of exploratory data analysis. The rationale behind this approach is to improve accessibility, allowing you, the reader, to delve into the areas that intrigue you the most. While the complete project remains available for those who prefer to explore it in a holistic environment, these individual sections offer a more specialized perspective on different facets of the data analysis project.

My project goes beyond the traditional scope of conducting an EDA, which is a standard process for experienced Data Analysts. The central idea here is to illustrate a comprehensive understanding of the process, specifically tailored to individuals with limited programming skills. Within these sections, you will discover a wealth of information, insights, and invaluable learning experiences. Each section is designed to stand independently while contributing to the overarching goal of demystifying data science.

Additionally, this project draws learnings from the activities within each section to identify activities that are iterable for any retail business. This contribution aims to develop a comprehensive strategic approach to conducting an EDA on Retail Businesses.

Whether you are a non-technical professional, a seasoned data analyst, a business development expert, or simply someone with a passion for data science, I have endeavored to dissect the process and activities to provide written summaries of key points, thought processes, rationale, and the necessity for each activity. This ensures that, regardless of your background, you will find valuable insights and inspiration throughout these pages.

The primary objective of this project is to bridge the knowledge gap for individuals with limited programming skills, offering step-by-step explanations and strategic insights into the realm of Exploratory Data Analysis.

The sections are structured as follows

Section 1: Retail Business EDA - Inspect, Prepare, Process Data

Section 2: Retail Business EDA using SQL: Item Category Data (Current Notebook)

Section 3: Retail Business EDA using SQL: Loss Rate Percentage Data

Section 4: Retail Business EDA using SQL: Wholesale Data

Section 5: Retail Business EDA using SQL: Transactions Data

Section 6: Retail Business EDA: SQL Joins - 4 Sales Datasets

(

I encourage you to explore these sections in an order that best aligns with your interests, needs, and level of expertise. I hope that the project inspires and equips you with the knowledge and enthusiasm to propel your journey in the world of data analytics and international development.

Thank you for embarking on this data-driven adventure with me. I wish you a rewarding and insightful journey through the sectional project.

About the Author

```
# Let's start by importing the datasets into the python notebook environment.
# Query Cell: Data Import from Multiple Files
# Summary:
# In this query cell, four datasets—'veg_category_df,' 'veg_txn_df,' 'veg_whsle_df,' and 'loss_rate_df'—are imported from separate CSV files.
# These datasets contain information related to supermarket sales and associated data, which will be used for further analysis and exploratory data analysis (EDA) tasks.
# Importing these datasets is the initial step in preparing the data for analysis, and they will be explored, cleaned, and examined in subsequent steps to extract insights and make data-driven decisions.

veg_category_df = pd.read_csv(r'/kaggle/input/supermarket-sales-data/annex1.csv')
```

```
veg_txn_df = pd.read_csv(r'/kaggle/input/supermarket-sales-data/annex2.csv')
veg_whsle_df = pd.read_csv(r'/kaggle/input/supermarket-sales-data/annex3.csv')
loss_rate_df = pd.read_csv(r'/kaggle/input/supermarket-sales-data/annex4.csv')
```

Inspect, Prepare, and Process Data using Python Notebook

Introduction

This first section of the project lays the essential foundation for meaningful data analysis. The first and foundational phase of any data analysis endeavor is the preparation and processing of the dataset. It is during this crucial stage that the raw data is inspected, refined, organized, and formatted making it conducive to deriving valuable insights and conducting in-depth analysis.

The section focuses on data inspection, understanding data types, standardizing column names, cleaning data types and formatting dates, identifying patterns and anomalies, and iterative data inspection. It aims to enhance data quality, consistency, and data integrity while maintaining a clear record of the process.

The strategic approach employed in this section ensures that the dataset is not only processed but also refined, organized, and optimized for insightful analysis, setting the stage for robust and data-driven decision-making. We lay the groundwork for a methodical and effective exploratory data analysis (EDA). This process ensures that our data is not just processed but refined, organized, and optimized for insightful analysis, setting the stage for robust and data-driven decision-making.

NB: Elaborate narratives have also been included in markdown above each query cells or as comments within each query cell to assist in following along with the queries.
Beginning of Section 1 activities with query 1 below.

Query 1: Initial Data Inspection

The initial step involves a quick glance at the first rows of the dataset. This serves as a "meet and greet" with the data, enabling us to understand its structure at a high level. By observing the initial entries, we can gain insights into its format and contents.

Query 1: Inspect the first few rows of a dataframe called `veg_category_df` to gain an overview of its columns and values.

veg_category_df.head()

	Item Code	Item Name	Category Code	Category Name
0	102900005115168	Niushou Shengcai	1011010101	Flower/Leaf Vegetables
1	102900005115199	Sichuan Red Cedar	1011010101	Flower/Leaf Vegetables
2	102900005115625	Local Xiaomao Cabbage	1011010101	Flower/Leaf Vegetables
3	102900005115748	White Caitai	1011010101	Flower/Leaf Vegetables
4	102900005115762	Amaranth	1011010101	Flower/Leaf Vegetables

Query 2: Data Type Inspection

- Inspecting data types and column names is essential for understanding the dataset's structure.
- By examining the data types, we can differentiate between numerical values, text, and date entries.
- Additionally, understanding column names enhances our awareness of what each piece of data represents.

Query 2: Define Query to Conduct an EDA on a dataframe to examine the data types and column names.

veg_category_df.dtypes

Using .dtypes() method in python generates the data types for each column as seen below.

```
Item Code      int64
Item Name      object
```

Category Code int64

Category Name object

dtype: object

Query 3: Column Name Standardization

Query 3 implements learnings from query 2 where we queried datatypes and columns for each column in the 'veg_category_df' and we learned the datatypes are fine but the column names can be cleaned to remove spaces between words.

Relevance

- Data consistency is paramount.
- By standardizing column names for clarity and uniformity, we facilitate smoother and more comprehensible data analysis.
- Clear column names are essential for readability and interpretation.

Activities

1. The column name 'Item Code' has been updated to 'item_code' using snake case.
2. The column name 'Item Name' has been transformed to 'item_name' using snake case.
3. The column name 'Category Code' has been modified to 'category_code' using snake case.
4. The column name 'Category Name' has been adjusted to 'category_name' using snake case.

In [7]:

```
# Query 3: assuming the current column names are 'old_name_1', 'old_name_2', etc and the new column names are 'new_name_1', 'new_name_2', etc.
new_column_names = ['item_code', 'item_name', 'category_code', 'category_name']

# rename the columns in place
veg_category_df.rename(columns=dict(zip(veg_category_df.columns, new_column_names)), inplace=True)

print(veg_category_df)
```

```
         item_code                         item_name  \
0    102900005115168                      Niushou Shengcai
1    102900005115199                     Sichuan Red Cedar
2    102900005115625                 Local Xiaomao Cabbage
3    102900005115748                         White Caitai
4    102900005115762                          Amaranth
..               ...                               ...
246  106958851400125              Haixian Mushroom (Bag) (4)
247  106971533450003               Haixian Mushroom (Bunch)
248  106971533455008              Haixian Mushroom (Bag) (3)
249  106973223300667  Chinese Caterpillar Fungus Flowers (Box) (2)
250  106973990980123           Hfyg Haixian Mushroom (Bunch)

     category_code         category_name
0       1011010101  Flower/Leaf Vegetables
1       1011010101  Flower/Leaf Vegetables
2       1011010101  Flower/Leaf Vegetables
3       1011010101  Flower/Leaf Vegetables
4       1011010101  Flower/Leaf Vegetables
..             ...                   ...
246     1011010801        Edible Mushroom
247     1011010801        Edible Mushroom
248     1011010801        Edible Mushroom
249     1011010801        Edible Mushroom
250     1011010801        Edible Mushroom

[251 rows x 4 columns]
```

Query 4: Check dataset columns and values to gain overview

veg_category_df.head()

	item_code	item_name	category_code	category_name
0	102900005115168	Niushou Shengcai	1011010101	Flower/Leaf Vegetables

1	102900005115199	Sichuan Red Cedar	1011010101	Flower/Leaf Vegetables
2	102900005115625	Local Xiaomao Cabbage	1011010101	Flower/Leaf Vegetables
3	102900005115748	White Caitai	1011010101	Flower/Leaf Vegetables
4	102900005115762	Amaranth	1011010101	Flower/Leaf Vegetables

Query 5: Identifying Patterns and Anomalies

In this phase, we begin exploring the dataset for patterns, trends, and anomalies. This step is vital for detecting irregularities or potential factors that may influence subsequent analyses. Customized queries and analysis techniques are employed to unveil these insights.

Query 5: Inspect the first few rows of a dataframe called `veg_txn_df` to gain an overview of its columns and values.

veg_txn_df.head()

	Date	Time	Item Code	Quantity Sold (kilo)	Unit Selling Price (RMB/kg)	Sale or Return	Discount (Yes/No)
0	2020-07-01	09:15:07.924	102900005117056	0.396	7.6	sale	No
1	2020-07-01	09:17:27.295	102900005115960	0.849	3.2	sale	No
2	2020-07-01	09:17:33.905	102900005117056	0.409	7.6	sale	No
3	2020-	09:19:45.	102900005115	0.421	10.0	sale	No

	07-01	450	823				
4	2020-07-01	09:20:23.686	102900005115908	0.539	8.0	sale	No

Query 6: Strategic Insights on activity: Data Cleaning and Handling Anomalies

Throughout the analysis, data quality issues and anomalies must be identified and addressed as they surface. This ensures data integrity and guarantees the accuracy of our analysis.

In [10]:

```
# Query 6: Conduct an Exploratory Data Analysis (EDA) on a dataframe to examine the data types and column names.
# A necessary step to understand the dataset's structure and prepare for subsequent data analysis tasks.
print(veg_txn_df.dtypes)
```

```
Date                        object
Time                        object
Item Code                    int64
Quantity Sold (kilo)       float64
Unit Selling Price (RMB/kg)    float64
Sale or Return              object
Discount (Yes/No)           object
dtype: object
```

Lessons from Query 6: Data Type and Column Name Adjustments for the 'veg_txn_df' Dataset

Modify the data type of the 'Date' column in 'veg_txn_df' from 'object' to 'Date'.

1. Revise the data type of the 'Time' column in 'veg_txn_df' from 'object' to 'Time'.
2. Transform the data type of the 'Discount' column in 'veg_txn_df' from 'object' to 'Boolean'.
3. Rename the column 'Item Code' in 'veg_txn_df' to 'item_code'.
4. Rename the column 'Quantity Sold (kilo)' in 'veg_txn_df' to 'quant_sold_kg'.

5. Rename the column 'Unit Selling Price (RMB/kg)' in 'veg_txn_df' to 'unit_selling_px_rmb/kg'.
6. Rename the column 'Sale or Return' in 'veg_txn_df' to 'sale/return'.
7. Rename the column 'Date' in 'veg_txn_df' to 'txn_date'.
8. Rename the column 'Time' in 'veg_txn_df' to 'txn_time'.

Query 7: Column Renaming

- Uniform and comprehensible column names are paramount to maintain consistency.
- Refining column names, when necessary, ensures that we maintain clarity and coherence throughout our analysis.
- In this step, we rename the columns of the 'veg_txn_df' dataset for better clarity and consistency.

Query 7 Activity

- We assume the current column names are 'old_name_1', 'old_name_2', etc and the new column names are 'txn_date', 'txn_time', 'item_code', 'qty_sold(kg)', 'unit_selling_px_rmb/kg','sale/return', 'discount(%)'.

In [11]:

```
# Query 7: Column Renaming:
# Create new_column_names dataframe with the desired column names on the table
new_column_names = ['txn_date', 'txn_time', 'item_code', 'qty_sold(kg)',
'unit_selling_px_rmb/kg','sale/return', 'discount(%)']

# rename the columns in place
veg_txn_df.rename(columns=dict(zip(veg_txn_df.columns, new_column_names)), inplace=True)

print(veg_txn_df)
```

```
      txn_date     txn_time       item_code  qty_sold(kg)  \
0     2020-07-01  09:15:07.924  102900005117056         0.396
1     2020-07-01  09:17:27.295  102900005115960         0.849
2     2020-07-01  09:17:33.905  102900005117056         0.409
3     2020-07-01  09:19:45.450  102900005115823         0.421
```

```
4       2020-07-01  09:20:23.686  102900005115908        0.539
...          ...          ...             ...          ...
878498  2023-06-30  21:35:13.264  102900005115250        0.284
878499  2023-06-30  21:35:14.358  102900011022764        0.669
878500  2023-06-30  21:35:20.264  102900005115250        0.125
878501  2023-06-30  21:35:21.509  102900011016701        0.252
878502  2023-06-30  21:40:48.248  102900011022764        0.803

        unit_selling_px_rmb/kg sale/return discount(%)
0                         7.6        sale         No
1                         3.2        sale         No
2                         7.6        sale         No
3                        10.0         sale         No
4                         8.0        sale         No
...                       ...         ...         ...
878498                   24.0        sale         No
878499                   12.0        sale         No
878500                   24.0        sale         No
878501                    5.2         sale         No
878502                   12.0        sale         No

[878503 rows x 7 columns]
```

Query 8: Query Insights: Data Type Cleaning and Date Formatting

- Data preparation often includes cleaning data types and standardizing date formats.
- Consistent data types ensure that our analysis proceeds without interruptions, while properly formatted dates are critical for time-based analyses.

In this step, we clean the data type of the 'veg_txn_df' dataset and format the date for analysis:

- We convert the 'txn_date' column to the datetime data type.
- We add a new column 'day_of_week' with the formatted day of the week.
- We print the updated dataframe for analysis.

In [12]:

```
# Query 8:
# convert the date column to datetime datatype
veg_txn_df['txn_date'] = pd.to_datetime(veg_txn_df['txn_date'])

# add a new column with the formatted date
veg_txn_df['day_of_week'] = veg_txn_df['txn_date'].dt.strftime('%A')

# print the dataframe with the new column
print(veg_txn_df)
```

```
       txn_date     txn_time       item_code  qty_sold(kg)  \
0     2020-07-01  09:15:07.924  102900005117056         0.396
1     2020-07-01  09:17:27.295  102900005115960         0.849
2     2020-07-01  09:17:33.905  102900005117056         0.409
3     2020-07-01  09:19:45.450  102900005115823         0.421
4     2020-07-01  09:20:23.686  102900005115908         0.539
...          ...           ...             ...           ...
878498 2023-06-30  21:35:13.264  102900005115250         0.284
878499 2023-06-30  21:35:14.358  102900011022764         0.669
878500 2023-06-30  21:35:20.264  102900005115250         0.125
878501 2023-06-30  21:35:21.509  102900011016701         0.252
878502 2023-06-30  21:40:48.248  102900011022764         0.803

       unit_selling_px_rmb/kg sale/return discount(%) day_of_week
0                         7.6        sale          No   Wednesday
1                         3.2        sale          No   Wednesday
2                         7.6        sale          No   Wednesday
3                        10.0        sale          No   Wednesday
4                         8.0        sale          No   Wednesday
...                       ...         ...         ...         ...
```

878498	24.0	sale	No	Friday
878499	12.0	sale	No	Friday
878500	24.0	sale	No	Friday
878501	5.2	sale	No	Friday
878502	12.0	sale	No	Friday

[878503 rows x 8 columns]

Query 9: Iterative Data Inspection

Data inspection isn't a one-time affair; it's an iterative process. Revisiting the initial data inspection as needed throughout the analysis keeps our understanding fresh and aligned with the data's evolving context.

Query 9: Check dataset columns and values to gain overview

veg_txn_df.head()

	txn_date	txn_time	item_code	qty_sold(kg)	unit_selling_px_rmb/kg	sale/return	discount(%)	day_of_week
0	2020-07-01	09:15:07.924	102900005117056	0.396	7.6	sale	No	Wednesday
1	2020-07-01	09:17:27.295	102900005115960	0.849	3.2	sale	No	Wednesday
2	2020-07-01	09:17:33.905	102900005117056	0.409	7.6	sale	No	Wednesday
3	2020-07-01	09:19:45.450	102900005115823	0.421	10.0	sale	No	Wednesday
4	2020-07-01	09:20:23.686	102900005115908	0.539	8.0	sale	No	Wednesday

Query 10: Data Type and Column Name Cleaning

In this step, we clean the data types and column names of the 'veg_txn_df' dataset for our final project presentation: 1) We change the data type of 'txn_date' to 'Date' for proper date representation. 2) We change the data type of 'txn_time' to 'Time' for accurate time representation. 3) We change the data type of 'discount(%)' to 'Boolean' to represent discounts as True/False. 4) We adjust 'qty_sold' to handle negative quantities, indicating returns correctly.

In [14]:

```
# Query 10: Data Type and Column Name Cleaning
veg_txn_df = veg_txn_df.astype({'txn_time': 'timedelta64[s]','discount(%)': 'bool'})
veg_txn_df['txn_date'] = pd.to_datetime(veg_txn_df['txn_date'])
veg_txn_df['txn_date'] = veg_txn_df['txn_date'].dt.floor('D')
veg_txn_df['txn_time'] = veg_txn_df['txn_time'].dt.total_seconds()
veg_txn_df['txn_time'] = pd.to_datetime(veg_txn_df['txn_time'],unit='s').dt.strftime('%H:%M:%S')
```

```
veg_txn_df.head()
```

	txn_date	txn_time	item_code	qty_sold(kg)	unit_selling_px_rmb/kg	sale/return	discount(%)	day_of_week
0	2020-07-01	09:15:07	102900005117056	0.396	7.6	sale	True	Wednesday
1	2020-07-01	09:17:27	102900005115960	0.849	3.2	sale	True	Wednesday
2	2020-07-01	09:17:33	102900005117056	0.409	7.6	sale	True	Wednesday
3	2020-07-01	09:19:45	102900005115823	0.421	10.0	sale	True	Wednesday
4	2020-07-01	09:20:23	102900005115908	0.539	8.0	sale	True	Wednesday

Query 11: Analysis of Negative 'qty_sold(kg)' Values in Relation to 'Sale/Return' Data

1. The examination revealed that the presence of negative values in the 'qty_sold(kg)' column is primarily associated with product returns.
2. The total count of both rows and columns in the dataset remains consistent at 461 rows and 8 columns.
3. This analysis offers a more profound comprehension of the anomaly represented by negative 'qty_sold(kg)' values within the dataset.

In [15]:

```
# Query 11: Understanding Negative Quantity Sold Values
# In this step, we explore the negative values in the 'qty_sold(kg)' column within the 'veg_txn_df'
transaction table to identify why qty sold is negative, and identify trends and correlations with returned
items:

# 1) To identify the negative values in the 'qty_sold(kg)' column, use the condition
veg_txn_df['qty_sold(kg)'] < 0.
negative_qty = veg_txn_df[veg_txn_df['qty_sold(kg)'] < 0]

# 2) To focus on cases where items are returned (sale/return == 'return') and have negative quantities, use
veg_txn_df[(veg_txn_df['qty_sold(kg)'] < 0) & (veg_txn_df['sale/return'] == 'return').
negative_qty_returned = veg_txn_df[(veg_txn_df['qty_sold(kg)'] < 0) & (veg_txn_df['sale/return'] ==
'return')]

print(negative_qty.describe())
print(negative_qty_returned.describe())
```

```
txn_date     item_code  qty_sold(kg)
count                        461  4.610000e+02    461.000000
mean   2022-01-11 10:18:28.893709568  1.030405e+14     -0.650588
min              2020-07-01 00:00:00  1.029000e+14     -9.082000
25%              2021-09-20 00:00:00  1.029000e+14     -1.000000
50%              2021-10-13 00:00:00  1.029000e+14     -0.489000
75%              2022-10-07 00:00:00  1.029000e+14     -0.318000
max              2023-06-08 00:00:00  1.069715e+14     -0.025000
std                              NaN  7.419623e+11      0.650815

       unit_selling_px_rmb/kg
count              461.000000
```

```
mean                 9.004338
min                  1.900000
25%                  5.200000
50%                  7.200000
75%                 10.000000
max                100.000000
std                  6.763735
                        txn_date     item_code  qty_sold(kg)  \
count                        461  4.610000e+02    461.000000
mean   2022-01-11 10:18:28.893709568  1.030405e+14     -0.650588
min              2020-07-01 00:00:00  1.029000e+14     -9.082000
25%              2021-09-20 00:00:00  1.029000e+14     -1.000000
50%              2021-10-13 00:00:00  1.029000e+14     -0.489000
75%              2022-10-07 00:00:00  1.029000e+14     -0.318000
max              2023-06-08 00:00:00  1.069715e+14     -0.025000
std                              NaN  7.419623e+11      0.650815

       unit_selling_px_rmb/kg
count              461.000000
mean                 9.004338
min                  1.900000
25%                  5.200000
50%                  7.200000
75%                 10.000000
max                100.000000
std                  6.763735
```

Query 12: Conduct an Exploratory Data Analysis (EDA) on a dataframe to examine the data types and column names.

```
# A necessary step to understand the dataset's structure and prepare for subsequent data analysis tasks.
veg_txn_df.dtypes
```

txn_date datetime64[ns]

txn_time object

item_code int64

qty_sold(kg) float64

unit_selling_px_rmb/kg float64

sale/return object

discount(%) bool

day_of_week object

dtype: object

Query 13: Inspect the first few rows of a dataframe called `veg_whsle_df` to gain an overview of its columns and values.

veg_whsle_df.head()

	Date	Item Code	Wholesale Price (RMB/kg)
0	2020-07-01	102900005115762	3.88
1	2020-07-01	102900005115779	6.72
2	2020-07-01	102900005115786	3.19
3	2020-07-01	102900005115793	9.24
4	2020-07-01	102900005115823	7.03

Query 14: Exploratory Data Analysis for Data Type and Column Name Inspection

Conducting an Exploratory Data Analysis (EDA) on the dataframe to examine data types and column names is a necessary preliminary step to gain a thorough understanding of the dataset's structure, facilitating the preparation for subsequent data analysis tasks.

Inspect Data Types and Column Names

```
print(veg_whsle_df.dtypes)
```

```
Date                        object
Item Code                    int64
Wholesale Price (RMB/kg)    float64
dtype: object
```

Query 14: Learnings from data type and column name inspection:

1. The 'Date' column is of the 'object' datatype.
2. All column names are represented as strings.

Query 15: Required Data Cleaning Actions

1. Change the data type of veg_whsle_df['Date'] from 'object' to 'Date'.
2. Rename the 'Date' column to 'whsle_date'.
3. Rename the 'Wholesale Price (RMB/kg)' column to 'whsle_px_rmb-kg'.

```
# Query 15: Data Type Enhancement and Date Simplification
# In the process of cleaning the 'veg_whsle_df' dataset, we undertake essential data type adjustments and date simplification to improve the dataset's utility and clarity for analytical purposes.

# Data Type Enhancement:
# 1) Convert the data type of the 'Date' column in 'veg_whsle_df' from 'Object' to 'Date'.
veg_whsle_df['Date'] = pd.to_datetime(veg_whsle_df['Date'])

# Date Simplification:
# 2) Truncate the 'Date' column to retain only the date component without the time information.
veg_whsle_df['Date'] = veg_whsle_df['Date'].dt.floor('D')

# Inspect Data Types After Modifications
print(veg_whsle_df.dtypes)
```

Results:
1) The 'Date' column is now of the 'Date' data type, facilitating accurate date-based operations.

```
Date                        datetime64[ns]
Item Code                         int64
Wholesale Price (RMB/kg)          float64
dtype: object
```

Query 16: Column Name Standardization

As part of the data cleansing process for the 'veg_whsle_df' dataset, we undertake vital actions to enhance column names, aiming to ensure consistency and clarity in data representation.

Column Name Enhancement:
1) Change the column name 'Date' to 'whsle_date'.
2) Alter the column name 'Wholesale Price (RMB/kg)' to 'whsle_px_rmb-kg'.

```
# Define New Column Names
new_column_names = ['whsle_date', 'item_code', 'whsle_px_rmb-kg']

# Apply Column Name Changes
veg_whsle_df.rename(columns=dict(zip(veg_whsle_df.columns, new_column_names)), inplace=True)

# Display the Updated DataFrame
print(veg_whsle_df.head())
```

```
 whsle_date       item_code  whsle_px_rmb-kg
0 2020-07-01  102900005115762             3.88
1 2020-07-01  102900005115779             6.72
2 2020-07-01  102900005115786             3.19
3 2020-07-01  102900005115793             9.24
4 2020-07-01  102900005115823             7.03
```

Query 17: Inspect the first few rows of a dataframe called `loss_rate_df` to gain an overview of its columns and values.

loss_rate_df.head

```
<bound method NDFrame.head of          Item Code                         Item Name  \
0    102900005115168                    Niushou Shengcai
1    102900005115199                   Sichuan Red Cedar
2    102900005115250                Xixia Black Mushroom (1)
3    102900005115625                 Local Xiaomao Cabbage
4    102900005115748                      White Caitai
..               ...                               ...
246  106971533455008              Haixian Mushroom (Bag) (3)
247  106971563780002                 Xianzongye (Bag) (2)
248  106972776821582                 Xianzongye (Bag) (3)
249  106973223300667  Chinese Caterpillar Fungus Flowers (Box) (2)
250  106973990980123            Hfyg Haixian Mushroom (Bunch)

     Loss Rate (%)
0             4.39
1            10.46
2            10.80
3             0.18
4             8.78
..             ...
246           1.30
247           0.00
248           9.43
249          11.13
250           0.12


[251 rows x 3 columns]>
```

In [22]:

```
# Query 18: Data Type and Column Name Examination
```

In the initial phase of the Exploratory Data Analysis (EDA) process, we inspect the data types and column names within the 'loss_rate_df' dataset to gain a comprehensive understanding of its structure.
This process of refining column names is a critical step in the data preparation process for effective data analysis and presentation, ultimately enhancing the quality and clarity of the 'loss_rate_df' dataset.

```
# Inspect Data Types and Column Names
print(loss_rate_df.dtypes)
```

```
Item Code        int64
Item Name       object
Loss Rate (%)  float64
dtype: object
```

Query 18: Learnings from exploration data type and column name.

Following an initial exploratory data analysis (EDA) query on 'loss_rate_df,' we have identified specific actions to enhance the dataset's clarity and consistency.

Query 19: Implementing learnings from query 20.

1. Modify the column name 'Item Code' to 'item_code.'
2. Modify the column name 'Item Name' to 'item_name.'
3. Modify the column name 'Loss Rate (%)' to 'lossrate%.'

In [23]:

```
# Query 19: Column Name Refinement
# In the process of cleaning the 'loss_rate_df' dataset, we undertake essential actions to enhance the column names, aiming to ensure consistency and clarity in data representation.

# Column Name Enhancement:
# 1) Modify the column name 'Item Code' to 'item_code.'
# 2) Modify the column name 'Item Name' to 'item_name.'
```

```
# 3) Modify the column name 'Loss Rate (%)' to 'loss_rate_%.'

# Define New Column Names
new_column_names = ['item_code', 'item_name', 'loss_rate_%']

# Apply Column Name Changes
loss_rate_df.rename(columns=dict(zip(loss_rate_df.columns, new_column_names)), inplace=True)

# Display the Updated DataFrame
print(loss_rate_df)

# Results from EDA Query
# The dataset 'loss_rate_df' has undergone these column name alterations for improved data representation and analysis.
```

	item_code	item_name	loss_rate_%
0	102900005115168	Niushou Shengcai	4.39
1	102900005115199	Sichuan Red Cedar	10.46
2	102900005115250	Xixia Black Mushroom (1)	10.80
3	102900005115625	Local Xiaomao Cabbage	0.18
4	102900005115748	White Caitai	8.78
..	...	...	...
246	106971533455008	Haixian Mushroom (Bag) (3)	
247	106971563780002	Xianzongye (Bag) (2)	
248	106972776821582	Xianzongye (Bag) (3)	
249	106973223300667	Chinese Caterpillar Fungus Flowers (Box) (2)	
250	106973990980123	Hfyg Haixian Mushroom (Bunch)	

246	1.30
247	0.00
248	9.43
249	11.13
250	0.12

[251 rows x 3 columns]

In [24]:

```
# Drawing lessons from section 1 activities
```

Strategic Insights: Inspect, Prepare, and Process Data

Drawing lessons project activities.

In the realm of data analysis and preparation, a strategic and standardized approach can significantly enhance the accessibility and understanding of data for individuals with limited coding experience. This approach is designed to be iterative and can serve as a foundation for the "Data Exploration and Initial Data Processing" section.

Section Introduction

Every step of data preparation and processing is meticulously documented. This documentation creates a clear and organized record of the entire process. These records serve as a reference, allowing for transparency and reproducibility in our analysis.

The objective of this section is to equip individuals with a systematic approach to exploring and preparing datasets for analysis. We will employ a step-by-step process to ensure that data is readily accessible and understandable. This approach is designed to be repeatable for various datasets, making it a valuable tool for individuals new to data analysis.

Here is a detailed context for each of the activities in the strategic approach for Data Exploration and Initial Data Processing in the context of an Exploratory Data Analysis (EDA) process:

Activity 1: Initial Data Inspection

- **Description:** This activity involves looking at the first few rows of the dataset to get a preliminary understanding of its structure. By displaying the initial rows, you can quickly see what columns are available and what kind of data is present.
- **Rationale:** The initial data inspection helps you understand the structure of the data, identify potential data quality issues, and get a sense of the dataset's content.
- **Queries in this project:**
    - veg_category_df.head(): This query displays the first few rows of the veg_category_df dataframe.
    - veg_txn_df.head(): Similar to the previous query but for the veg_txn_df dataframe.
    - veg_whsle_df.head(): Displays the initial rows of the veg_whsle_df dataframe.
    - loss_rate_df.head(): Provides a preview of the initial rows of the loss_rate_df.

Activity 2: Data Type Inspection

- **Description:** In this activity, you inspect the data types of columns in the dataset. Understanding data types is crucial because it affects how you perform calculations and operations on the data.
- **Rationale:** This step helps you understand the dataset's structure and ensures that data types are appropriate for analysis.
- **Queries in this project:**
    - veg_category_df.dtypes: This query displays the data types of columns in the veg_category_df.
    - veg_txn_df.dtypes: Similar to the previous query but for the veg_txn_df.
    - veg_whsle_df.dtypes: Shows the data types of columns in the veg_whsle_df.
    - loss_rate_df.dtypes: Displays the data types of columns in the loss_rate_df.

Activity 3: Column Name Standardization

- **Description:** This activity focuses on renaming columns in a standardized format (e.g., snake case) to enhance clarity and consistency.
- **Rationale:** Standardized column names improve data readability and maintain consistency across datasets.
- **Queries in this project:**
    - Renaming columns in each dataframe to follow a consistent naming convention, such as snake case.

Activity 4: Data Type Cleaning and Date Formatting

- **Description:** This activity involves cleaning data types and formatting date columns for analysis. For example, converting date columns to datetime format and extracting additional date-related information.
- **Rationale:** Cleaning data types and formatting dates ensure data consistency and prepare the data for date-based analysis.
- **Queries in this project:**
    - For veg_txn_df, you convert the date column to datetime format and add a new column for the day of the week.
    - For veg_whsle_df, you convert the date column to datetime format and remove time information.

Activity 5: Investigating Negative Values (Data Anomaly Detection)

- **Description:** This activity is about exploring and analyzing negative values in specific columns to understand their significance. In this case, you are investigating trends related to negative values, such as returns.
- **Rationale:** Identifying and understanding negative values, like returns, is important for data quality and analysis.
- **Queries in this project:** (You would perform specific queries to identify and analyze negative values, such as returns, using appropriate functions and methods for your dataset.)

In this strategic approach, these activities provide a systematic and standardized way to inspect, prepare, and clean datasets before conducting EDA. Each activity is designed to enhance data accessibility, quality,

and consistency, making it a valuable tool for individuals, especially those new to data analysis, to follow in their EDA process.

In [25]:

Conclusion:

Conclusion: Inspect, Prepare, and Process Data

In this first section of our comprehensive retail business exploratory data analysis (EDA) project, we embarked on a journey to lay the essential foundation for meaningful data analysis. Data preparation and processing represent the foundational phase of any data analysis endeavor, where the raw data is meticulously inspected, refined, organized, and formatted to pave the way for insightful analysis.

Our focus was on understanding the data structure, identifying potential data quality issues, and preparing the data for a robust EDA. The strategic approach employed in this section ensured that the dataset was not only processed but also refined, organized, and optimized for insightful analysis, setting the stage for data-driven decision-making.

Throughout this section, we systematically explored various facets of data preparation, which included activities such as initial data inspection, data type inspection, column name standardization, data type cleaning, date formatting, and investigating negative values. Each of these activities played a crucial role in enhancing data accessibility, quality, and consistency.

The systematic documentation of every step taken during this section serves as a valuable reference, offering transparency and reproducibility in our analysis. Whether you are a novice or an experienced data analyst, the activities presented here provide a structured and standardized approach to inspecting, preparing, and cleaning datasets, making data accessible and understandable.

As you progress to the subsequent sections of this project, you'll be well-equipped with a strong foundation to dive deeper into the world of retail data analysis. Whether your interests lie in category management, pricing strategies, profitability analysis, quality control, or inventory management, the strategic insights gained in this section will be instrumental in your journey.

I encourage you to explore the remaining sections in an order that aligns with your interests and expertise. I hope this first section has provided you with valuable insights and inspiration as you continue your exploration of data analytics for retail businesses.

Thank you for joining me in this data-driven adventure, and I look forward to your continued journey through the remaining sections of the project.

Collaboration and Engagement:

If you're inspired by this journey and have insights to share, we invite you to fork this notebook and contribute your perspectives. Engage with us through the comments section to provide feedback, share opinions, and offer valuable advice. Together, we can enhance this notebook and expand its horizons, demystifying data science and making it more accessible to all.

Let's Work Together:

Beyond contributing to this notebook, we're open to collaborating on data projects. If you have a project in mind or need data analysis support, don't hesitate to reach out. Whether it's a collaborative effort or data-driven solutions for your business, we're here to assist. Contact us through the provided email or LinkedIn for inquiries and discussions.