University of Westminster
School of Computer Science and Engineering
5COSC020W Database Systems

Module Leader: Ms. Yaalini Balathasan

Individual Coursework

Name: W.A.Heshan Peiris

UOW No.: w1999712

IIT No.: 20221171

Table Of Contents

Contents

1) Objective

Design and implement a data pipeline that extracts data from a data source, performs data transformation, and loads it into a destination storage system. This task simulates a common data engineering scenario.

2) Scenario

You work for a retail company, and your task is to create a data pipeline to process and store customer purchase data or inventory management data. The data comes from multiple CSV Biles and needs to be transformed before being loaded into a relational database.

3) Tasks

3.1) Data Extraction

I created a small dataset with information about customers' sales transactions and inventory for a retail store, including features like customer ID, customer Name, email in customer table. Product ID, product Name, product Quantity in inventory table. Transaction ID, Card Type, Transaction Date in Sales Transactions table.

3.2) Data Exploration

(a) Number of data points.

Inventory: 1000 Rows

```
In [105]: # (a) Number of data points of Inventory table
          num_data_points_in_the_Inventory_table = Inventory.shape[0]
          print(f"num_data_points_in_the_Inventory_table: {num_data_points_in_the_Inventory_table}\n")

          num_data_points_in_the_Inventory_table: 1000
```

Customers: 1000 Rows

```
In [104]: # (a) Number of data points of Customer table
          num_data_points_in_the_Customer_table = Customers.shape[0]
          print(f"num_data_points_in_the_Customer_table: {num_data_points_in_the_Customer_table}\n")

          num_data_points_in_the_Customer_table: 1000
```

Sales Transactions: 1000 Rows

```
In [106]: # (a) Number of data points of Sales_Transactions table
          num_data_points_in_the_Sales_Transactions_table = Sales_Transactions.shape[0]
          print(f"num_data_points_in_the_Sales_Transactions_table: {num_data_points_in_the_Sales_Transactions_table}\n")

          num_data_points_in_the_Sales_Transactions_table: 1000
```

(b) Name of attributes.

Inventory:

Product ID

Product Name

Last Restock Date

Product Quantity

Unit Price

Supplier

In [73]:
```
# (b) Name of attributes of Inventory table
attributes_name_Inventory_table = Inventory.columns
print(f"attributes_name_of_Inventory_table: {attributes_name_Inventory_table}\n")
```

```
attributes_name_of_Inventory_table: Index(['product_id', 'product_name', 'last_restock_date', 'product_quantity',
       'unit_price', 'supplier'],
      dtype='object')
```

Customers:

Customer ID

Customer Name

Age

Email

Address

Phone Number

Income

In [72]:
```
# (b) Name of attributes of Customer table
attributes_name_of_Customer_table = Customers.columns
print(f"attributes_name_of_Customer_table: {attributes_name_of_Customer_table}\n")
```

```
attributes_name_of_Customer_table: Index(['customer_id', 'customer_name', 'age', 'email', 'address',
       'phone_number', 'income'],
      dtype='object')
```

Sales Transactions:

Transaction ID

Customer ID

Product ID

Quantity

Transaction Date

Total Amount

Card Type

In [74]:
```
# (b) Name of attributes of Sales_Transactions table
attributes_name_Sales_Transactions_table = Sales_Transactions.columns
print(f"attributes_name_of_Sales_Transactions_table: {attributes_name_Sales_Transactions_table}\n")
```

```
attributes_name_of_Sales_Transactions_table: Index(['transaction_id', 'customer_id', 'product_id', 'quantity',
       'transaction_date', 'total_amount', 'Card_Type'],
      dtype='object')
```

(c) Type of attributes

Inventory:

Product ID - **Integer**

Product Name - **String**

Last Restock Date - **Date**

Product Quantity - **Integer**

Unit Price - **Float**

Supplier – **String**

In [75]:
```
# (c) Type of attributes of Customer table
attribute_types_of_Customer_table = Customers.dtypes
print("attribute_types_of_Customer_table:")
print(attribute_types_of_Customer_table)
```

```
attribute_types_of_Customer_table:
customer_id       object
customer_name     object
age              float64
email             object
address           object
phone_number      object
income           float64
dtype: object
```

Customers:

Customer ID - **Integer**

Customer Name - **String**

Age - **Integer**

Email - **String**

Address - **String**

Phone Number - **Integer**

Income – **Float (Currency)**

In [76]:
```
# (c) Type of attributes of Inventory table
attribute_types_of_Inventory_table = Inventory.dtypes
print("attribute_types_of_Inventory_table:")
print(attribute_types_of_Inventory_table)
```

```
attribute_types_of_Inventory_table:
product_id            object
product_name          object
last_restock_date     object
product_quantity     float64
unit_price           float64
supplier              object
dtype: object
```

Sales Transactions:

Transaction ID **- Integer**

Customer ID - **Integer**

Product ID **- Integer**

Quantity - **Integer**

Transaction Date - **Date**

Total Amount **- Float (Currency)**

Card Type – **String**

```
In [77]: # (c) Type of attributes of Sales_Transactions table
         attribute_types_of_Sales_Transactions_table = Sales_Transactions.dtypes
         print("attribute_types_of_Sales_Transactions_table:")
         print(attribute_types_of_Sales_Transactions_table)

         attribute_types_of_Sales_Transactions_table:
         transaction_id       object
         customer_id          object
         product_id           object
         quantity            float64
         transaction_date     object
         total_amount          int64
         Card_Type            object
         dtype: object
```

(d) Number of missing values for each attribute

Inventory:

Product ID - **0**

Product Name - **0**

Last Restock Date - **109**

Product Quantity - **110**

Unit Price - **102**

Supplier – **0**

In [114]:
```
# (d) Number of missing values for each attribute
missing_values_of_Inventory_table = Inventory.isnull().sum()
print("\nNumber of missing values for each attribute:")
print(missing_values_of_Inventory_table)
```

```
Number of missing values for each attribute:
product_id            0
product_name          0
last_restock_date   109
product_quantity    110
unit_price          102
supplier              0
dtype: int64
```

Customers:

Customer ID - **0**

Customer Name - **0**

Age - **109**

Email - **114**

Address - **101**

Phone Number - **0**

Income – **257**

```
In [113]: # (d) Number of missing values for each attribute
missing_values_of_Customer_table = Customers.isnull().sum()
print("\nNumber of missing values for each attribute:")
print(missing_values_of_Customer_table)
```

```
Number of missing values for each attribute:
customer_id        0
customer_name      0
age              109
email            114
address          101
phone_number       0
income           257
dtype: int64
```

Sales Transactions:

Transaction ID - **0**

Customer ID - **0**

Product ID - **114**

Quantity - **112**

Transaction Date - **108**

Total Amount - **0**

Card Type – **276**

```
In [80]: # (d) Number of missing values for each attribute
missing_values_of_Sales_Transactions_table =Sales_Transactions.isnull().sum()
print("\nNumber of missing values for each attribute:")
print(missing_values_of_Sales_Transactions_table)
```

```
Number of missing values for each attribute:
transaction_id        0
customer_id           0
product_id          114
quantity            112
transaction_date    108
total_amount          0
Card_Type           276
dtype: int64
```

(e) Entry errors for each attribute

In [116]:
```python
# Get the entry errors for each attribute
entry_errors = (Customers.apply(pd.unique).apply(len) - 1).to_list()
entry_errors
```

Out[116]:
```
[999, 999, 73, 886, 899, 999, 743]
```

In [117]:
```python
# Get the entry errors for each attribute
entry_errors = (Inventory.apply(pd.unique).apply(len) - 1).to_list()
entry_errors
```

Out[117]:
```
[999, 817, 338, 853, 898, 999]
```

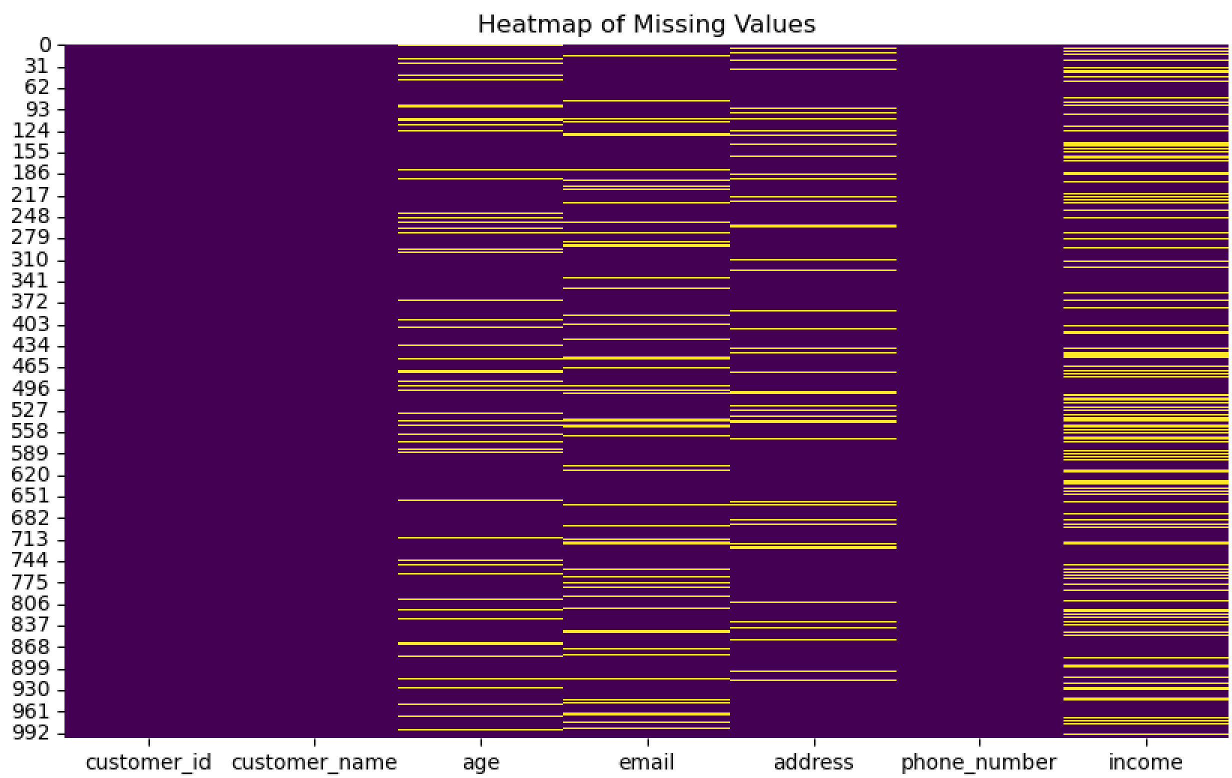
In [118]:
```python
# Get the entry errors for each attribute
entry_errors = (Sales_Transactions.apply(pd.unique).apply(len) - 1).to_list()
entry_errors
```

Out[118]:
```
[999, 999, 886, 847, 330, 998, 16]
```

(f) Heatmaps to check missing values

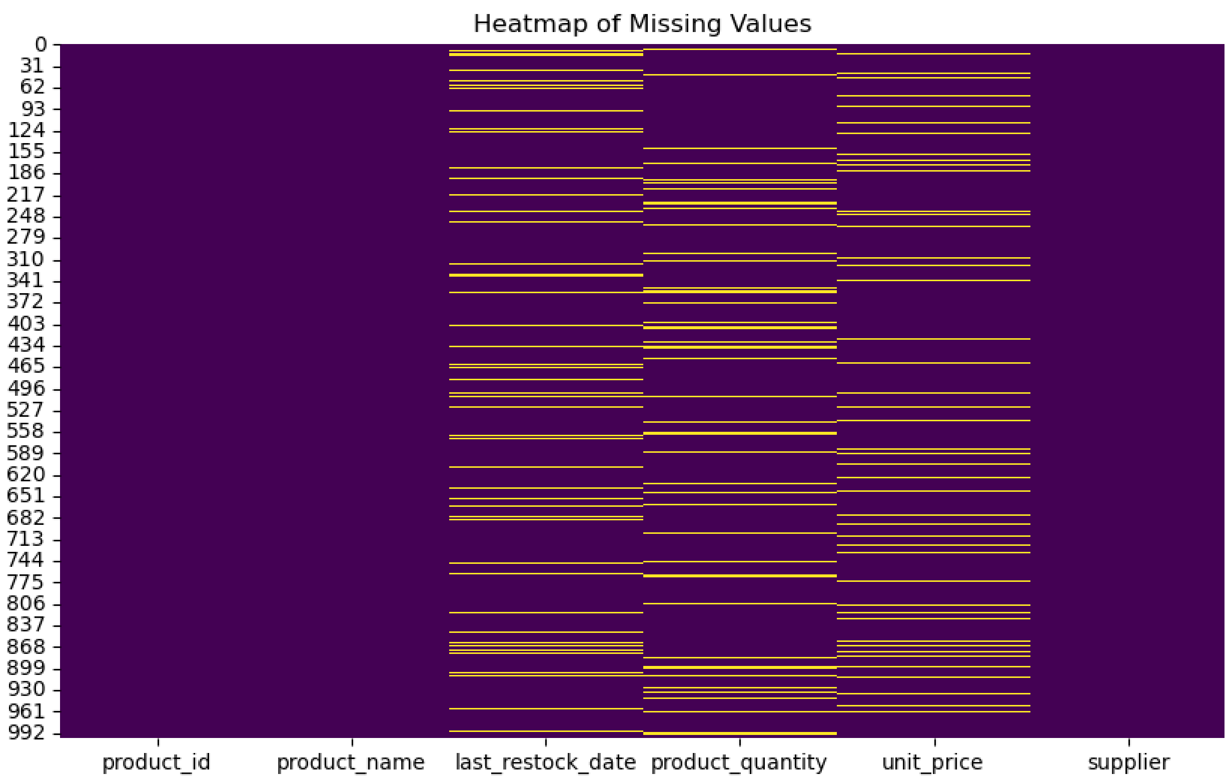
Customer Table

In [82]:
```
# (f) Heatmap to check missing values
plt.figure(figsize=(10, 6))
sns.heatmap(Customers.isnull(), cbar=False, cmap='viridis')
plt.title('Heatmap of Missing Values')
plt.show()
```



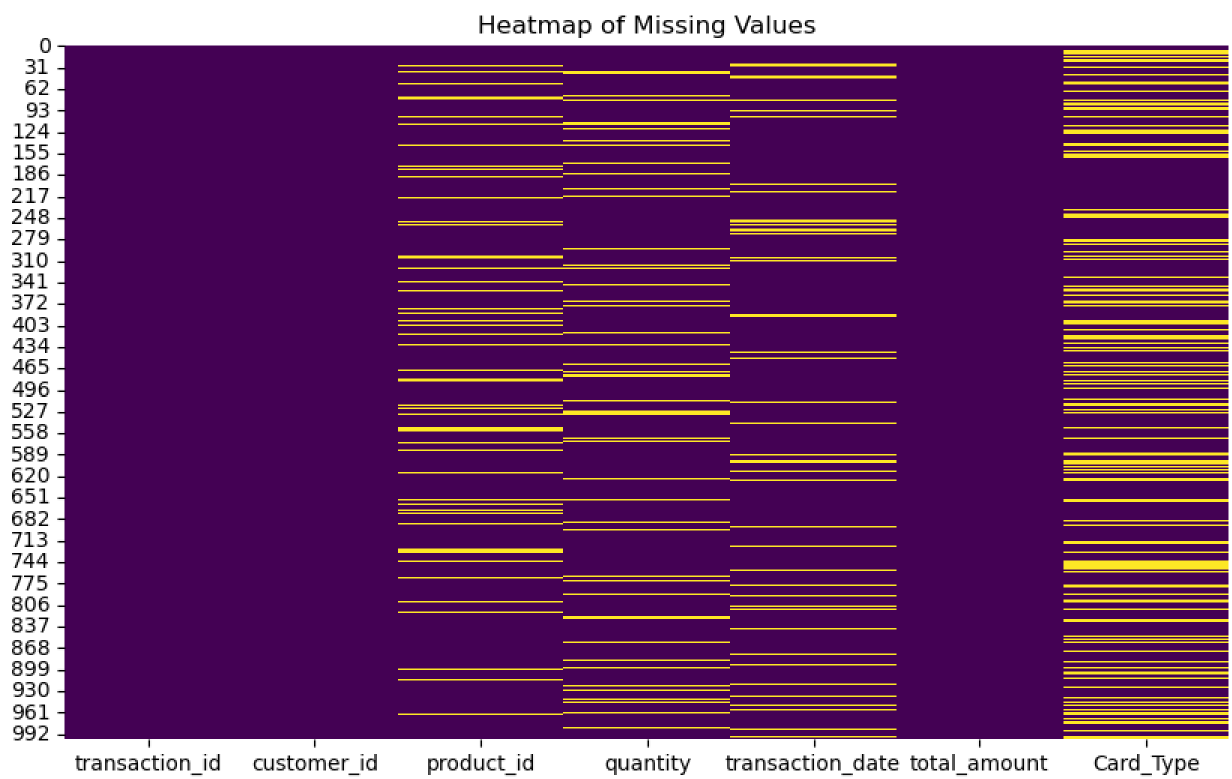
Inventory Table

In [83]:
```
# (f) Heatmap to check missing values
plt.figure(figsize=(10, 6))
sns.heatmap(Inventory.isnull(), cbar=False, cmap='viridis')
plt.title('Heatmap of Missing Values')
plt.show()
```



Sales Transactions Table

In [85]:
```
# (f) Heatmap to check missing values
plt.figure(figsize=(10, 6))
sns.heatmap(Sales_Transactions.isnull(), cbar=False, cmap='viridis')
plt.title('Heatmap of Missing Values')
plt.show()
```



3.3) Data Transformation

Part a

Integrating data from multiple CSV Biles of the initial data source into a single dataset. Eg.combining purchase data from different months, years, etc.

[211]:
```
#3.3(a) Integrating data from multiple CSV Biles of the initial data source into a single dataset
import pandas as pd

#Reading the 3 data tables from the CSV file
Customers = pd.read_csv('Customers.csv')
Inventory = pd.read_csv('Inventory.csv')
Sales_Transactions = pd.read_csv('Sales_Transactions.csv')
```

[212]:
```
#Merging the transactions
Stocks = Sales_Transactions.merge(Customers, on='customer_id')

# droping any duplicates
Stocks = merged_data.select_dtypes(include=['object', 'int64', 'float64']).drop_duplicates()

#Saving the merged data to a new CSV file
merged_data.to_csv('Stocks.csv', index=False)
```

[213]:
```
#Loading the merged data
Orders = pd.read_csv('Stocks.csv')

#Displaying the merged data
print(Stocks)
```

```
    transaction_id customer_id product_id_x  quantity transaction_date  \
0       13-7427019  87-2919006   46-2732354    3952.0       10.06.2023
1       65-2968490  56-9876147   07-4959214     318.0       15.02.2023
2       93-7941794  73-2941955   69-8808573    6494.0       27.01.2023
3       41-5559823  83-9114467          NaN    9650.0       27.05.2023
4       59-6523445  42-9271721   33-1257533    1821.0       14.01.2023
..             ...         ...          ...       ...              ...
995     04-3287363  88-3114549   19-3860523    7349.0       04.09.2023
996     06-8190016  43-7099027   36-0923368    6155.0              NaN
997     54-3489274  00-8254353   34-8540914    7498.0              NaN
998     30-4499724  93-5885871   07-3399157    6729.0       24.11.2022
999     76-9064451  31-1340840          NaN    4837.0       09.09.2023

     total_amount           Card_Type product_id_y  \
0          136314       visa-electron   46-2732354
1          478949                 jcb   07-4959214
2          217955                 jcb   69-8808573
3          111286                 NaN          NaN
4          166472                 jcb   33-1257533
..            ...                 ...          ...
995        431431      china-unionpay   19-3860523
996        289473                 NaN   36-0923368
997         47864                 NaN   34-8540914
998        266238                 NaN   07-3399157
999         17636  diners-club-enroute          NaN
```

```
                       product_name last_restock_date  product_quantity  \
0                 Onions - Spanish        31.01.2023            5905.0
1               Scotch - Queen Anne        18.12.2022            2021.0
2          Wine - Pinot Noir Latour        28.06.2023            7634.0
3               Arizona - Green Tea        11.09.2023            8355.0
4      Gingerale - Schweppes, 355 Ml        24.08.2023            6522.0
..                              ...               ...               ...
995                Foil - Round Foil        12.09.2023              81.0
996                     Sauce - Mint        16.01.2023            7804.0
997        Chivas Regal - 12 Year Old        05.03.2023               NaN
998                      Lemon Grass        11.03.2023            9226.0
999  Cheese - Havarti, Roasted Garlic        16.02.2023               NaN

     unit_price             supplier
0       4220.30    Fleurette Fernier
1       3464.97        Clio Le Grove
2       1955.62            Elmer Bea
3        268.38     Anstice Wycliffe
4       1663.12          Trudy Landy
..          ...                  ...
995     2887.52        Maud Harcarse
996     4008.53  Alessandra Farnill
997     1114.83     Beaufort Cusiter
998     3452.85           Alys Prium
999     1597.30           Yance Feak

[1000 rows x 13 columns]
```

Part b

(b) Handling missing values by either removing rows with missing data or inputting values.

In [214]:
```
import pandas as pd

#Reading the merged data
Stocks = pd.read_csv('Stocks.csv')
```

In [215]:
```
#Checking the number of rows with missing values
rows_with_null_values = merged_data.isnull().sum()
print(rows_with_null_values)

#Removing rows with missing values
Stocks.dropna(inplace=True)

#Checking the number of rows after removing missing values
print(len(Stocks))
```

```
transaction_id         0
customer_id            0
product_id_x         114
quantity             112
transaction_date     108
total_amount           0
Card_Type            276
product_id_y         114
product_name           0
last_restock_date    109
product_quantity     110
unit_price           102
supplier               0
dtype: int64
365
```

In [216]:
```
#Saving the cleaned data to a new csv file
Stocks.to_csv('Stocks_cleaned.csv', index=False)

#Reading the cleaned orders data from csv file
Stocks_cleaned = pd.read_csv('Stocks_cleaned.csv')

#Displaying the new cleaned orders csv data
print(Stocks_cleaned)
```

```
    transaction_id customer_id product_id_x  quantity transaction_date  \
0       13-7427019  87-2919006   46-2732354    3952.0       10.06.2023
1       65-2968490  56-9876147   07-4959214     318.0       15.02.2023
2       93-7941794  73-2941955   69-8808573    6494.0       27.01.2023
3       59-6523445  42-9271721   33-1257533    1821.0       14.01.2023
4       98-9652722  75-4953322   56-9365892    6566.0       13.02.2023
..             ...         ...          ...       ...              ...
360     19-6566776  48-0186542   99-8744602    7725.0       04.11.2023
361     34-5432532  75-9927939   40-8196289    5418.0       01.11.2023
362     24-9838707  39-9332634   11-0825981     728.0       06.06.2023
363     73-3038659  75-1150384   69-8539974    9314.0       15.01.2023
364     04-3287363  88-3114549   19-3860523    7349.0       04.09.2023

     total_amount        Card_Type product_id_y  \
0          136314    visa-electron   46-2732354
1          478949              jcb   07-4959214
2          217955              jcb   69-8808573
3          166472              jcb   33-1257533
4           43692              jcb   56-9365892
..            ...              ...          ...
360        126922  americanexpress   99-8744602
361        231479              jcb   40-8196289
362        289006       mastercard   11-0825981
363         40067          maestro   69-8539974
364        431431   china-unionpay   19-3860523
```

```
                      product_name last_restock_date  product_quantity  \
0                Onions - Spanish        31.01.2023            5905.0   
1               Scotch - Queen Anne        18.12.2022            2021.0   
2          Wine - Pinot Noir Latour        28.06.2023            7634.0   
3     Gingerale - Schweppes, 355 Ml        24.08.2023            6522.0   
4                 Hinge W Undercut        14.04.2023            2700.0   
..                             ...               ...               ...   
360                  Water - Tonic        29.04.2023            1970.0   
361                  Rice - Basmati        29.04.2023            7235.0   
362             Pop - Club Soda Can        02.05.2023            3961.0   
363       Bread - Mini Hamburger Bun        26.04.2023            2226.0   
364               Foil - Round Foil        12.09.2023              81.0   

     unit_price             supplier  
0       4220.30   Fleurette Fernier  
1       3464.97       Clio Le Grove  
2       1955.62           Elmer Bea  
3       1663.12         Trudy Landy  
4       3761.92     Rafaelita Fippe  
..          ...                 ...  
360     2147.33       Marketa Maffy  
361      713.01     Wilbur Fishpoole  
362     3145.23       Genni Persian  
363     3486.25  Charity McCarrison  
364     2887.52       Maud Harcarse  

[365 rows x 13 columns]
```

Part c

(c) Removing columns of redundant features.

In [217]:
```
#3.3(c)
import pandas as pd

#Loading the cleaned CSV file into a pandas DataFrame
Stocks_cleaned = pd.read_csv('Stocks_cleaned.csv')
```

In [218]:
```
#Identifying the columns to remove
columns_to_remove = ['Card_Type','total_amount']

#Removing the unwanted columns
Stocks_cleaned.drop(columns=columns_to_remove, axis=1, inplace=True)
```

In [219]:
```
#Saving the updated DataFrame to a new CSV file
Stocks_cleaned.to_csv('Updated_data.csv', index=False)

#Reading the cleaned Stocks data
Updated_data = pd.read_csv('Updated_data.csv')

#Displaying the new cleaned Stocks csv data
print(Updated_data)
```

```
    transaction_id customer_id product_id_x  quantity transaction_date  \
0       13-7427019  87-2919006   46-2732354    3952.0       10.06.2023   
1       65-2968490  56-9876147   07-4959214     318.0       15.02.2023   
2       93-7941794  73-2941955   69-8808573    6494.0       27.01.2023   
3       59-6523445  42-9271721   33-1257533    1821.0       14.01.2023   
4       98-9652722  75-4953322   56-9365892    6566.0       13.02.2023   
..             ...         ...          ...       ...              ...   
360     19-6566776  48-0186542   99-8744602    7725.0       04.11.2023   
361     34-5432532  75-9927939   40-8196289    5418.0       01.11.2023   
362     24-9838707  39-9332634   11-0825981     728.0       06.06.2023   
363     73-3038659  75-1150384   69-8539974    9314.0       15.01.2023   
364     04-3287363  88-3114549   19-3860523    7349.0       04.09.2023   

    product_id_y                  product_name last_restock_date  \
0     46-2732354              Onions - Spanish        31.01.2023   
1     07-4959214           Scotch - Queen Anne        18.12.2022   
2     69-8808573        Wine - Pinot Noir Latour        28.06.2023   
3     33-1257533  Gingerale - Schweppes, 355 Ml        24.08.2023   
4     56-9365892              Hinge W Undercut        14.04.2023   
..           ...                           ...               ...   
360   99-8744602                 Water - Tonic        29.04.2023   
361   40-8196289                 Rice - Basmati        29.04.2023   
362   11-0825981            Pop - Club Soda Can        02.05.2023   
363   69-8539974     Bread - Mini Hamburger Bun        26.04.2023   
364   19-3860523              Foil - Round Foil        12.09.2023   
```

```
     product_quantity  unit_price            supplier
0              5905.0     4220.30   Fleurette Fernier
1              2021.0     3464.97       Clio Le Grove
2              7634.0     1955.62           Elmer Bea
3              6522.0     1663.12         Trudy Landy
4              2700.0     3761.92     Rafaelita Fippe
..                ...         ...                 ...
360            1970.0     2147.33       Marketa Maffy
361            7235.0      713.01     Wilbur Fishpoole
362            3961.0     3145.23       Genni Persian
363            2226.0     3486.25  Charity McCarrison
364              81.0     2887.52       Maud Harcarse

[365 rows x 11 columns]
```

Part d

(c) Aggregating purchase data, such as calculating total sales per customer.

```
In [220]: #3.3(d) Aggregating purchase data
#Creating a new column named Total Amount which depicts the Total by multiplying Unit price by Quantity

import pandas as pd

#Reading the cleaned orders data
Updated_data = pd.read_csv('Updated_data.csv')

#Creating a column named 'Total Expenditure' and calculating the Total
Updated_data['Total Expenditure'] = Updated_data['product_quantity'] * Updated_data['unit_price']
```

```
In [221]: #Saving the updated dataframe to a new CSV file
Updated_data.to_csv('Updated_data_with_total_amount.csv', index=False)

#Reading the updated dataframe with total Total Expenditure
Updated_data_with_total_amount = pd.read_csv('Updated_data_with_total_amount.csv')

#Displaying the updated dataframe with total Total Expenditure
print(Updated_data_with_total_Total Expenditure)
```

```
    transaction_id customer_id product_id_x  quantity transaction_date  \
0       13-7427019  87-2919006   46-2732354    3952.0       10.06.2023
1       65-2968490  56-9876147   07-4959214     318.0       15.02.2023
2       93-7941794  73-2941955   69-8808573    6494.0       27.01.2023
3       59-6523445  42-9271721   33-1257533    1821.0       14.01.2023
4       98-9652722  75-4953322   56-9365892    6566.0       13.02.2023
..             ...         ...          ...       ...              ...
360     19-6566776  48-0186542   99-8744602    7725.0       04.11.2023
361     34-5432532  75-9927939   40-8196289    5418.0       01.11.2023
362     24-9838707  39-9332634   11-0825981     728.0       06.06.2023
363     73-3038659  75-1150384   69-8539974    9314.0       15.01.2023
364     04-3287363  88-3114549   19-3860523    7349.0       04.09.2023

    product_id_y                product_name last_restock_date  \
0     46-2732354           Onions - Spanish        31.01.2023
1     07-4959214          Scotch - Queen Anne       18.12.2022
2     69-8808573      Wine - Pinot Noir Latour     28.06.2023
3     33-1257533  Gingerale - Schweppes, 355 Ml     24.08.2023
4     56-9365892           Hinge W Undercut        14.04.2023
..           ...                          ...               ...
360   99-8744602              Water - Tonic        29.04.2023
361   40-8196289             Rice - Basmati        29.04.2023
362   11-0825981          Pop - Club Soda Can      02.05.2023
363   69-8539974    Bread - Mini Hamburger Bun     26.04.2023
364   19-3860523            Foil - Round Foil      12.09.2023


     product_quantity  unit_price            supplier  Total Expenditure
0              5905.0     4220.30   Fleurette Fernier        24920871.50
1              2021.0     3464.97       Clio Le Grove         7002704.37
2              7634.0     1955.62           Elmer Bea        14929203.08
3              6522.0     1663.12         Trudy Landy        10846868.64
4              2700.0     3761.92     Rafaelita Fippe        10157184.00
..                ...         ...                 ...                ...
360            1970.0     2147.33       Marketa Maffy         4230240.10
361            7235.0      713.01    Wilbur Fishpoole         5158627.35
362            3961.0     3145.23       Genni Persian        12458256.03
363            2226.0     3486.25  Charity McCarrison         7760392.50
364              81.0     2887.52       Maud Harcarse          233889.12

[365 rows x 12 columns]
```

3.4 Self-reflection

Difficulties

One of the main difficulties I encountered when creating my data pipeline was understanding the different data sources and how they fit together. There were many different data sources, each with its own format and structure, and it was difficult to figure out how to integrate them all into a single pipeline. To assure the quality of the data, I also had to create cleaning and filtering processes because the data was usually incorrect or incomplete.

Another challenge I faced was debugging and troubleshooting the pipeline. There were many points where the pipeline failed, and it was often difficult to identify the source of the problem. This required me to carefully examine the data and the code to find the errors.

Learning

Despite the difficulties, building my data pipeline taught me a lot. The value of design and planning was among the most crucial lessons I took away. I took some time to arrange the various steps of the procedure and the equipment I would need before I began constructing the pipeline. This assisted me in avoiding errors and in building a scalable and effective process.

I also learned a lot about data cleaning and transformation. I developed several techniques for cleaning and transforming the data, such as removing duplicates, handling missing values, and normalizing the data. These techniques were essential for ensuring the quality of the data and for preparing it for analysis.

Along with the technical skills I acquired, I also improved my critical thinking and problem-solving abilities. I gained knowledge on how to tackle challenging issues by decomposing them into smaller, more doable tasks. I also gained knowledge on how to evaluate data critically and spot any issues.

Overall, I found the experience of creating my data pipeline to be challenging but rewarding. I learned a lot about data wrangling and data analysis, and I developed valuable skills that will be useful in my future career.