In [ ]:

In [ ]:
```
"""
Introduction:

This project shows how Python is used to analyze customers' orders in internet sales. This analysis uses basic data structures such as lists,
dictionaries, tuples, and sets. The structures are used to process and reveal business intelligence. The workflow inside the business includes
classifying customers with their spending habits, identifying the most purchased products, evaluating category-level sales performance, and highlighting
cross category purchasing trends.

The outcome of this project is a comprehensive report that summarizes both customer classifications and revenue distribution. This report also includes
marketing information, customer retention strategies, and inventory management. This end to end analyzed project reflects how Basic Python techniques
are used to solve real world business problems which require key skills for data analysts.

"""
```

1. Store customer orders

In [ ]:
```
""" Create a list of customer names """
```

In [141…
```
#  Create a list of customer names

customers_names = ["Usman", "Dax", "Hari", "Bella", "Drew", "Paul", "Ryan", "Ram", "Steven",
                   "Thomas", "Prakash", "Victoria", "Alex", "Chandra", "Ella"]

# Count and print the number of customers
print(len(customers_names))
```
```
15
```

In [49]:
```
""" Store each customer's order details (customer name, product, price, category) as  tuples inside a list """
```
Out[49]:
```
" Store each customer's order details (customer name, product, price, category) as  tuples inside a list "
```

In [143…
```
# Generate a dataset named as "customer_Orders" that contains all customer purchases.
# Each order has a tuple with customer_name, product, price, category.
# Example: ("Usman", "Printer", 220, "Electronics") = (customer_name, product, price, category)

customer_Orders = [
    ("Usman", "Printer", 220, "Electronics"),
    ("Usman", "T-shirt", 20, "Clothing"),
    ("Dax", "Boots", 40, "Clothing"),
    ("Dax", "Gloves", 16, "Home_Essentials"),
    ("Hari", "Laptop", 772, "Electronics"),
    ("Hari", "Sweater", 72, "Clothing"),
    ("Bella", "Desktop_Computer", 650, "Electronics"),
    ("Drew", "Coat", 41, "Clothing"),
    ("Paul", "Mattress", 99, "Home_Essentials"),
    ("Paul", "Headphones", 44, "Electronics"),
    ("Ryan", "TV", 250, "Electronics"),
    ("Ram", "Shorts", 33, "Clothing"),
    ("Steven", "Smartwatch", 200, "Electronics"),
    ("Thomas", "Couch", 365, "Home_Essentials"),
    ("Thomas", "Bluetooth_Speaker", 45, "Electronics"),
    ("Prakash", "Jeans", 45, "Clothing"),
    ("Victoria", "Trash_Can", 24, "Home_Essentials"),
    ("Alex", "Tablet", 243, "Electronics"),
    ("Chandra", "Polo_Shirt", 36, "Clothing"),
    ("Ella", "Earbuds", 90, "Electronics"),
    ("Ella", "Mixer", 47, "Home_Essentials"),
    ("Ella", "Towels", 90, "Clothing"),
    ("Usman", "Laptop", 240, "Electronics")
]

# Print the total number of customer orders in the dataset
print(len(customer_Orders))
```
```
23
```

In [53]:
```
"""  Use a dictionary where keys are customer names and values are lists of ordered products """
```
Out[53]:
```
'  Use a dictionary where keys are customer names and values are lists of ordered products '
```

In [145…
```
# Change the dataset into the dictionary format{key:value}. key is a customer name. A value is a tuple with (customer_name, product, price, category)
# Example: "Ella": [("Earbuds", 90, "Electronics"), ("Mixer", 47, "Home_Essentials"), ("Towels", 90, "Clothing")]
# Iterate each tuple in customer_Orders.
# Initialize "customer_names_values" using setdefault(), and append the (product, price, category) into the "customer_names_values".

customer_names_values = {}   #
for customer_name, product, price, category in customer_Orders:
    customer_names_values.setdefault(customer_name, []).append((product, price, category))

# Print the dictionary, customer_names_values
print(customer_names_values)
```
```
{'Usman': [('Printer', 220, 'Electronics'), ('T-shirt', 20, 'Clothing'), ('Laptop', 240, 'Electronics')], 'Dax': [('Boots', 40, 'Clothing'), ('Gloves', 16, 'Home_Essentials')], 'Hari': [('Laptop', 772, 'Electronics'), ('Sweater', 72, 'Clothing')], 'Bella': [('Desktop_Computer', 650, 'Electronics')], 'Drew': [('Coat', 41, 'Clothing')], 'Paul': [('Mattress', 99, 'Home_Essentials'), ('Headphones', 44, 'Electronics')], 'Ryan': [('TV', 250, 'Electronics')], 'Ram': [('Shorts', 33, 'Clothing')], 'Steven': [('Smartwatch', 200, 'Electronics')], 'Thomas': [('Couch', 365, 'Home_Essentials'), ('Bluetooth_Speaker', 45, 'Electronics')], 'Prakash': [('Jeans', 45, 'Clothing')], 'Victoria': [('Trash_Can', 24, 'Home_Essentials')], 'Alex': [('Tablet', 243, 'Electronics')], 'Chandra': [('Polo_Shirt', 36, 'Clothing')], 'Ella': [('Earbuds', 90, 'Electronics'), ('Mixer', 47, 'Home_Essentials'), ('Towels', 90, 'Clothing')]}
```

2. Classify products by category

In [58]:
```
"""  Use a dictionary to map each product to its respective category  """
```
Out[58]:
```
'  Use a dictionary to map each product to its respective category  '
```

In [147…
```
# Make a dictionary named product_categories which maps each product into its corresponding category.
# Examples:  "Printer": "Electronics", "Boots": "Clothing"

product_categories = {
    "Printer": "Electronics",
    "Boots": "Clothing",
    "Desktop_Computer": "Electronics",
    "Coat": "Clothing",
    "Mattress": "Home_Essentials",
    "TV": "Electronics",
    "Shorts": "Clothing",
    "Smartwatch": "Electronics",
    "Couch": "Home_Essentials",
    "Jeans": "Clothing",
    "Trash_Can": "Home_Essentials",
    "Tablet": "Electronics",
    "Polo_Shirt": "Clothing",
    "T-shirt": "Clothing",
    "Gloves": "Home_Essentials",
    "Laptop": "Electronics",
    "Sweater": "Clothing",
    "Headphones": "Electronics",
    "Bluetooth_Speaker": "Electronics",
    "Earbuds": "Electronics",
    "Mixer": "Home_Essentials",
    "Towels": "Clothing",
    "Laptop": "Electronics"
}

#  Print the length of dictionary (product_categories)
print(len(product_categories))
```
```
22
```

In [62]:
```
"""  Create a set of unique product categories  """
```
Out[62]:
```
'  Create a set of unique product categories  '
```

In [149…
```
# Find product categories from product_categories, and then generate a set of Unique product_categories.

Unique_Product_Categories = set(product_categories.values())
print("\n   A set of unique product categories:",  Unique_Product_Categories )
```
```
  A set of unique product categories: {'Clothing', 'Home_Essentials', 'Electronics'}
```

In [66]:
```
"""  Display all available product categories  """
```
Out[66]:
```
'  Display all available product categories  '
```

In [151…
```
# Do iteration on Unique_Product_Categories using "for" loop
# Display each category.

for product_category in Unique_Product_Categories:
    print(" * ",  product_category)
```
```
*  Clothing
*  Home_Essentials
*  Electronics
```

3. Analyze customer orders

In [71]:
```
"""  Use a loop to calculate the total amount each customer spends  """
```
Out[71]:
```
'  Use a loop to calculate the total amount each customer spends  '
```

In [153…
```
# Using a "for" loop to calculate the total amount each customer spends.

# Create a dictionary to store the total amount spended by each customer
customer_total_amount_spending = {}

# Use the dictionary, customer_names_values that has keys (customer names) and values (product, price, category)
# Iterate this dictionary, and collect total amount per customer.
# All results stored in the dictionary, customer_total_amount_spending that has each customer name and total amount spending.

for customer_name, customer_orders in customer_names_values.items():
    total_amount = 0
    # Each customer order is a tuple that has (product, price, category)
    for product, price, category in customer_orders:
        total_amount += price
    customer_total_amount_spending[customer_name] = total_amount

# Total spending amount per customer

print("\n _____________-Total spending amount per customer____________")

for  customer_name, total_amount in customer_total_amount_spending.items():
    print(f"{ customer_name}: ${total_amount}")
```
```
 _____________-Total spending amount per customer____________
Usman: $480
Dax: $56
Hari: $844
Bella: $650
Drew: $41
Paul: $143
Ryan: $250
Ram: $33
Steven: $200
Thomas: $410
Prakash: $45
Victoria: $24
Alex: $243
Chandra: $36
Ella: $227
```

In [75]:
```
"""
*  If the total purchase value is above $100, classify the customer as a high-value buyer
*  If it is between $50 and $100, classify the customer as a moderate buyer
*  If it is below $50, classify them as a low-value buyer

"""
```
Out[75]:
```
'\n*  If the total purchase value is above $100, classify the customer as a high-value buyer \n*  If it is between $50 and $100, classify the customer as a moderate buyer \n*  If it is below $50, classify them as a low-value buyer\n \n'
```

In [155…
```
# Goals: Classify each customer to one of these categories (high-value buyer, moderate buyer, low-value buyer) under their total amount used.

# Store total spending amount for each customer in a dictionary
customer_spending_amount = {}

# Store customer classification under spending amount in a dictionary.
customer_classification_under_spending_amount = {}

# Calculate total spending for this customer

for customer_name, items in customer_names_values.items():
    # Find total spending from current customer
    # Unpack tuple
    total_spending_amount = sum(price for __, price, __ in items)

    customer_spending_amount[customer_name] = total_spending_amount

    # Use classification rules

    if total_spending_amount > 100:
        customer_classification_under_spending_amount[customer_name] = "high_Value_buyer"
    elif 50 <= total_spending_amount <= 100:
        customer_classification_under_spending_amount[customer_name] = "moderate_buyer"
    else:
        customer_classification_under_spending_amount[customer_name] = "low_value_buyer"

print(customer_spending_amount)

print("\n")

print(customer_classification_under_spending_amount)
```
```
{'Usman': 480, 'Dax': 56, 'Hari': 844, 'Bella': 650, 'Drew': 41, 'Paul': 143, 'Ryan': 250, 'Ram': 33, 'Steven': 200, 'Thomas': 410, 'Prakash': 45, 'Victoria': 24, 'Alex': 243, 'Chandra': 36, 'Ella': 227}


{'Usman': 'high_Value_buyer', 'Dax': 'moderate_buyer', 'Hari': 'high_Value_buyer', 'Bella': 'high_Value_buyer', 'Drew': 'low_value_buyer', 'Paul': 'high_Value_buyer', 'Ryan': 'high_Value_buyer', 'Ram': 'low_value_buyer', 'Steven': 'high_Value_buyer', 'Thomas': 'high_Value_buyer', 'Prakash': 'low_value_buyer', 'Victoria': 'low_value_buyer', 'Alex': 'high_Value_buyer', 'Chandra': 'low_value_buyer', 'Ella': 'high_Value_buyer'}
```

4. Generate business insights

In [80]:
```
"""  Calculate the total revenue per product category and store it in a dictionary """
```
Out[80]:
```
'  Calculate the total revenue per product category and store it in a dictionary '
```

In [157…
```
# Revenue per category

total_revenue_product_category = {}

# Go through each order in the customer_Orders using for loop.
# Each customer_Order has (customer_name, product, price, category).
for _, product, price, category in customer_Orders:    # Do not consider the customer name with using (_).
    total_revenue_product_category[category] = total_revenue_product_category.get(category, 0) + price

# Print the total revenue product category
print(total_revenue_product_category)
```
```
{'Electronics': 2754, 'Clothing': 377, 'Home_Essentials': 551}
```

In [84]:
```
"""  Extract unique products from all orders using a set  """
```
Out[84]:
```
'  Extract unique products from all orders using a set  '
```

In [159…
```
# Extract unique products from all orders using a set with a for loop

# for _, product, _, _ in  customer_Orders: Tuple decompose, and take product only.
unique_products = set(product for _, product, _, _ in  customer_Orders)

# print unique_products that has distinct product
print(unique_products)
```
```
{'Shorts', 'TV', 'T-shirt', 'Polo_Shirt', 'Earbuds', 'Laptop', 'Coat', 'Tablet', 'Mixer', 'Mattress', 'Trash_Can', 'Boots', 'Jeans', 'Couch', 'Bluetooth_Speaker', 'Sweater', 'Gloves', 'Towels', 'Printer', 'Desktop_Computer', 'Smartwatch', 'Headphones'}
```

In [88]:
```
"""  Use a list comprehension to find all customers who purchased electronics  """
```
Out[88]:
```
'  Use a list comprehension to find all customers who purchased electronics  '
```

In [161…
```
# Use a list comprehension to find all customers who purchased electronics

# list comprehension: Look at each Customer and corresponding tuple (product, price, category)
# Customers who purchased electronics
all_electronics_customers = [customer for customer, tuple in customer_names_values.items() if any(i == "Electronics" for _, _, i in tuple)]
print(all_electronics_customers)
```
```
['Usman', 'Hari', 'Bella', 'Paul', 'Ryan', 'Steven', 'Thomas', 'Alex', 'Ella']
```

In [96]:
```
"""Identify the top three highest-spending customers using sorting """
```
Out[96]:
```
'Identify the top three highest-spending customers using sorting '
```

In [163…
```
# Sort the dictionary (customer_total_amount_spending) with total_amount spending in decreasing order.
# key=lambda i: i[1] means look at the second member of the tuple
# reverse=True means look from the highest to lowest the total_amount spending
# [:3] means look at top three customers under
# Top 3 highest spending customers total_amount_spending
top_three_highest_spending_customers = sorted(customer_total_amount_spending.items(), key=lambda i: i[1], reverse=True)[:3]

print(top_three_highest_spending_customers)
```
```
[('Hari', 844), ('Bella', 650), ('Usman', 480)]
```

5. Organize and display data

In [101…
```
"""   Print a summary of each customer’s total spending and their classification """
```
Out[101…
```
'   Print a summary of each customer’s total spending and their classification '
```

In [165…
```
print("\nA summary of each customer’s total spending and their classification ")

print("\n")

for customer_name, total_spending in customer_total_amount_spending.items():
    print(f"{customer_name}: Spent ${total_spending} ---- {customer_classification_under_spending_amount[customer_name]}")
```
```
A summary of each customer’s total spending and their classification


Usman: Spent $480 ---- high_Value_buyer
Dax: Spent $56 ---- moderate_buyer
Hari: Spent $844 ---- high_Value_buyer
Bella: Spent $650 ---- high_Value_buyer
Drew: Spent $41 ---- low_value_buyer
Paul: Spent $143 ---- high_Value_buyer
Ryan: Spent $250 ---- high_Value_buyer
Ram: Spent $33 ---- low_value_buyer
Steven: Spent $200 ---- high_Value_buyer
Thomas: Spent $410 ---- high_Value_buyer
Prakash: Spent $45 ---- low_value_buyer
Victoria: Spent $24 ---- low_value_buyer
Alex: Spent $243 ---- high_Value_buyer
Chandra: Spent $36 ---- low_value_buyer
Ella: Spent $227 ---- high_Value_buyer
```

In [105…
```
"""   Use set operations to find customers who purchased from multiple categories   """
```
Out[105…
```
'   Use set operations to find customers who purchased from multiple categories   '
```

In [167…
```
# Store customers who purchase multiple categories.
customers_multiple_categories = []

for customer_name, customer_orders in customer_names_values.items():

    # Make a set of unique categories for each customer
    categories_set = {category for _, _, category in customer_orders}
    print(categories_set)

    if len(categories_set) > 1:
        customers_multiple_categories.append(customer_name)
        print(customers_multiple_categories)
        print("\n")

# print the customers who purchased from multiple categories

print("\n ----customers who purchased from multiple categories:-----")

print("\n")

for customer_name in customers_multiple_categories:
    print("-", customer_name)
```
```
{'Clothing', 'Electronics'}
['Usman']


{'Clothing', 'Home_Essentials'}
['Usman', 'Dax']


{'Clothing', 'Electronics'}
['Usman', 'Dax', 'Hari']


{'Electronics'}
{'Clothing'}
{'Home_Essentials', 'Electronics'}
['Usman', 'Dax', 'Hari', 'Paul']


{'Electronics'}
{'Clothing'}
{'Electronics'}
{'Home_Essentials', 'Electronics'}
['Usman', 'Dax', 'Hari', 'Paul', 'Thomas']


{'Clothing'}
{'Home_Essentials'}
{'Electronics'}
{'Clothing'}
{'Clothing', 'Home_Essentials', 'Electronics'}
['Usman', 'Dax', 'Hari', 'Paul', 'Thomas', 'Ella']



 ----customers who purchased from multiple categories:-----


- Usman
- Dax
- Hari
- Paul
- Thomas
- Ella
```

In [109…
```
"""   Identify common customers who bought both electronics and clothing   """
```
Out[109…
```
'   Identify common customers who bought both electronics and clothing   '
```

In [169…
```
common_customers_electronics_clothing = []

for customer_name, orders in customer_names_values.items():

    # A collection of customer's category.
    set_categories = {category for _, _, category in orders}

    if "Electronics" in set_categories and "Clothing" in set_categories:
        common_customers_electronics_clothing.append(customer_name)

# Print common customers who bought both electronics and clothing
print("\n---common customers who bought both electronics and clothing:---")

print("\n")

for customer_name in common_customers_electronics_clothing:
    print("*", customer_name)
```
```
---common customers who bought both electronics and clothing:---


* Usman
* Hari
* Ella
```

Actions

In [114…
```
"""
Customer order processing in python

1. Store customer order data using lists, tuples, and dictionaries

2. Retrieve and modify customer records using dictionary methods

"""
```
Out[114…
```
' \nCustomer order processing in python \n\n1. Store customer order data using lists, tuples, and dictionaries \n\n2. Retrieve and modify customer records using dictionary methods\n\n'
```

In [116…
```
#   1. Store customer order data using lists, tuples, and dictionaries
#  Already done with list, tuples, and dictionaries. I also include counts customers, counts orders, and groups ordders  into dictionaries.
```

In [118…
```
#  2. Retrieve and modify customer records using dictionary methods
"""
Here, I am going to retrieve, and modify customer records.
For examples:
Retrieve: customer_names_values.get("Ella")
Modify: customer_names_values["Ella"].append(("Short", 22, "Clothing"))
Remove: customer_names_values.pop("Ram")
  """
```
Out[118…
```
'\nHere, I am going to retrieve, and modify customer records. \nFor examples: \nRetrieve: customer_names_values.get("Ella")\nModify: customer_names_values["Ella"].append(("Short", 22, "Clothing"))\nRemove: customer_names_values.pop("Ram")\n  '
```

In [171…
```
# Retrieve a customer order (for eample, Ella's orders)

print("Ella's orders:", customer_names_values.get("Ella"))

print("\n")

# Modify (add)  Usman's orders

customer_names_values.setdefault("Usman", []).append(("Mouse", 27, "Electronics"))

print("Modify Usman's orders:", customer_names_values["Usman"] )

print("\n")

# Remove Ram's orders
remove_orders = customer_names_values.pop("Paul", "Mattress")
print("Remove Paul's orders:", remove_orders)
```
```
Ella's orders: [('Earbuds', 90, 'Electronics'), ('Mixer', 47, 'Home_Essentials'), ('Towels', 90, 'Clothing')]


Modify Usman's orders: [('Printer', 220, 'Electronics'), ('T-shirt', 20, 'Clothing'), ('Laptop', 240, 'Electronics'), ('Mouse', 27, 'Electronics')]


Remove Paul's orders: [('Mattress', 99, 'Home_Essentials'), ('Headphones', 44, 'Electronics')]
```

In [122…
```
"""

Classification and analysis

1. Use loops to categorize customers based on their total spending.

2. Use set operations to find common and unique products across different categories

"""
```
Out[122…
```
'\n\nClassification and analysis \n\n1. Use loops to categorize customers based on their total spending.\n\n2. Use set operations to find common and unique products across different categories\n\n'
```

In [124…
```
#  Use loops to categorize customers based on their total spending

# Already discussed the following points.

"""
* Loop through customer_names_values
* Total spending amount = sum(price for _, price, _ in customer_Orders)
* Classify into high_Value_buyer, moderate_buyer, low_value_buyer.

"""
```
Out[124…
```
'\n* Loop through customer_names_values\n* Total spending amount = sum(price for _, price, _ in customer_Orders)\n* Classify into high_Value_buyer, moderate_buyer, low_value_buyer. \n\n'
```

In [126…
```
# Use set operations to find common and unique products across different categories

# Already discussed this topic.
```

In [128…
```
# Insight generation
```

In [130…
```
"""
Insight generation
• Extract the high-value customers and most frequently purchased products
• Identify trends based on category-wise sales

"""
```
Out[130…
```
'\nInsight generation \n• Extract the high-value customers and most frequently purchased products \n• Identify trends based on category-wise sales \n\n'
```

In [173…
```
#  Extract the high-value customers

high_value_customers = [customer for customer, classification in customer_classification_under_spending_amount.items()
                        if classification == "high_Value_buyer"]
print(high_value_customers)


print("\n")

# Extract the most frequently purchased products


products_list = [product for _, product, _, _ in customer_Orders]

from collections import Counter
product_counts = Counter(products_list)

most_frequent_products = product_counts.most_common(5)

print(most_frequent_products)
```
```
['Usman', 'Hari', 'Bella', 'Paul', 'Ryan', 'Steven', 'Thomas', 'Alex', 'Ella']


[('Laptop', 2), ('Printer', 1), ('T-shirt', 1), ('Boots', 1), ('Gloves', 1)]
```

In [175…
```
# Identify trends based on category-wise sales
category_sales = {}
for _, product, price, category in customer_Orders:
    category_sales[category] = category_sales.get(category, 0) + price

print(category_sales)
```
```
{'Electronics': 2754, 'Clothing': 377, 'Home_Essentials': 551}
```

In [136…
```
"""
Result:
The final deliverable will be a detailed report summarizing customer classifications, total
sales per category, and key insights about purchase behavior. This project demonstrates
how Python’s data structures can be used to analyze real-world e-commerce data,
helping businesses make informed decisions.

"""
```
Out[136…
```
'\nResult: \nThe final deliverable will be a detailed report summarizing customer classifications, total \nsales per category, and key insights about purchase behavior. This project demonstrates \nhow Python’s data structures can be used to analyze real-world e-commerce data, \nhelping businesses make informed decisions.\n\n'
```

In [138…
```
"""
Course-End Project Analyzing Customer Orders Using Python

1. Customer Classification:

With the help of Python loops as well as conditionals, customers were divided into three categories under their total spending amount.
(a)     “high-value buyer” (more than $100),
(b)     “moderate buyer” (between $50 and $100),
(c)     “low-value buyer” (below $50,)
This classification indicates which customers support the most revenue.

2. Total Sales by Category:

Total revenue was calculated through all product categories such as Electronics, Clothing, and Home Essentials. Results provided:
(a)     Electronics made the highest overall sales
(b)     Clothing generated moderate sales.
(c)     Home Essentials showed the lower sales when compared to sales in Electronics.

3. Key Insights:

     (a) high-value customers: Many customers frequently purchased “Electronics”, showing higher revenue. To collect more revenue, these customers
         can be put as a higher priority.
     (b) highly purchased products: products such as laptops, headphones in electronics, and t-shirts, jeans in clothing seen frequently in the dataset.
     (c) category trends: Electronics was the main category for both revenue and frequent purchase. On the other hand, clothing products were purchased
         by many customers. Few customers purchased home essentials but still showed some revenue.
      (d) multiple categories purchased customers: Some customers purchased multiple categories, mainly electronics and clothing,.

4. Conclusion:

This project shows how Python’s data structures such as lists, dictionaries, and sets are applied into the internet sales data for further selling
data analysis. With classifying customers under category choice, and analyzing their product choice, businesses need to be considered the following
facts:
(a)     Maximize customer-oriented marketing efforts,
(b)     Improve product supply chain
(c)     Focus on retaining valuable customers

"""
```
Out[138…
```
'\nCourse-End Project Analyzing Customer Orders Using Python\n\n1. Customer Classification:\n\nWith the help of Python loops as well as conditionals, customers were divided into three categories under their total spending amount.\n(a)\t“high-value buyer” (more than $100),\n(b)\t“moderate buyer” (between $50 and $100),\n(c)\t“low-value buyer” (below $50,)\nThis classification indicates which customers support the most revenue. \n\n2. Total Sales by Category:\n\nTotal revenue was calculated through all product categories such as Electronics, Clothing, and Home Essentials. Results provided:\n(a)\tElectronics made the highest overall sales\n(b)\tClothing generated moderate sales.\n(c)\tHome Essentials showed the lower sales when compared to sales in Electronics.\n\n3. Key Insights:\n\n     (a) high-value customers: Many customers frequently purchased “Electronics”, showing higher revenue. To collect more revenue, these customers\n         can be put as a higher priority. \n     (b) highly purchased products: products such as laptops, headphones in electronics, and t-shirts, jeans in clothing seen frequently in the dataset. \n     (c) category trends: Electronics was the main category for both revenue and frequent purchase. On the other hand, clothing products were purchased\n         by many customers. Few customers purchased home essentials but still showed some revenue.\n      (d) multiple categories purchased customers: Some customers purchased multiple categories, mainly electronics and clothing,.\n\n4. Conclusion:\n\nThis project shows how Python’s data structures such as lists, dictionaries, and sets are applied into the internet sales data for further selling\ndata analysis. With classifying customers under category choice, and analyzing their product choice, businesses need to be considered the following\nfacts:\n(a)\tMaximize customer-oriented marketing efforts,\n(b)\tImprove product supply chain\n(c)\tFocus on retaining valuable customers \n\n'
```

In [ ]:

In [ ]: