**Ayush Meher,**

**Data Scientist,**

Experienced in analyzing Ecommerce Data, Marketing Data, CRM Data, Healthcare Dataset.

**Project:** We are having a tracking issue on this platform. We need to figure out the volume of orders this is affecting the total number of orders.

**Rule:**

Base Order: All the orders should have base order ending \_0,

Upselling Orders: All upselling orders should have -1\_0, -2\_0,…-9\_0 ending. <https://shop.organixx.com/products/multi-vita-maxx/?ref=nav>

Upselling generally involves moving the customer to a higher quantity or more premium version of the same product, which aligns with the tiered discount structure here.

Example: Accepted: Orders with No Issues.

A screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Rejected: Orders with Issues:

A screenshot of a computer

Description automatically generated

Objective:

Provide an analysis of how many orders are exhibiting this problem out of the total orders correctly processed, using the data provided above.

**Approach 1:**

Using Python Programming: Python is one of the most used language in analyzing big datasets.

**Pseudocode**

1. **Load Order Data from Excel**
   * Load data from an Excel file, where each row contains an Order ID.
2. **Initialize Storage for Orders**
   * Create an empty dictionary (order\_dict) to store all orders, categorized by their base order ID.
3. **Separate Base Orders and Upselling Orders**
   * For each Order ID:
     + Extract the base part of the order before any - or \_.
     + Store the Order ID under its base order in the dictionary (order\_dict).

*Example:*  
For Order ID "C93702301B-1\_0", the base order ID is "C93702301B". Store "C93702301B-1\_0" under "C93702301B" in order\_dict.

1. **Identify Missing Base Orders (\_0 suffix)**
   * For each base order in the dictionary:
     + Check if the base order includes an initial order ending in \_0.
     + If \_0 is missing, store this order in a separate dictionary (orders\_having\_issue).

*Example:*  
If "C93702301B\_0" is missing from the transactions under base order "C93702301B", add "C93702301B" to orders\_having\_issue.

1. **Calculate and Print Issues**
   * Print the list of problematic orders and their count.
   * Calculate the percentage of problematic orders out of all orders.
2. **Find Orders That Don’t End in \_0**
   * For each order in the dictionary:
     + Identify orders that don’t end in \_0 and store them in a list.

*Example:*  
If "C93702301B-1\_1" doesn’t end with \_0, add it to the orders\_not\_ending\_in\_0 list.

1. **Output and Save Results**
   * Print the problematic orders and orders that don’t end in \_0.
   * Save the orders\_having\_issue to a CSV file for reporting.

Output:

'D4587E914A': ['D4587E914A-4\_0'], '543D480FE6': ['543D480FE6-1\_0']}

Total problematic orders: 3549

Out of 30174 orders, 11.76% have issues

Data successfully written to D:\Outlier AI\marketing analysis\orders\_with\_issues.csv

Orders not ending in '\_0': ['AAB517E3A1\_3']

Total orders not ending in '\_0': 1

Approach 2:

**Using Excel:** <https://docs.google.com/spreadsheets/d/1jMdOoL66a3xGbUYlwJEhVbAhlgE_KTI1PJHc-brLE_A/edit?usp=sharing>

1. **Identify Relevant Orders:**

* Filter the dataset to focus only on orders that should have the base identifier "\_0" and exclude those with other sequences like "-2\_0", "-3\_0", etc.
* Use the **LEFT function** to extract the base part of the order IDs (before any suffix like -1\_0, -2\_0).

2. Calculate **Total Orders:**

* Determine the total number of orders in the dataset.

3. **Extract Unique Order Numbers:**

* Use a **Pivot Table** to summarize the data and extract unique order IDs, ensuring we only analyze distinct orders.

4. Verify **Correct Base Sequence:**

* For each unique order, check if it has the correct base sequence (i.e., ending in "\_0").
* Use **IF(COUNTIF(...))** to check if an order ID with the correct base sequence exists. If it does, return "T", indicating the presence of the correct base sequence, otherwise return "F".

5. Calculate **Orders Missing Base Sequence:**

* Subtract the number of orders with the correct base sequence "\_0" from the total number of orders to identify those that are missing it.

