B2B Courier Charges Accuracy Analysis

Introduction

ABC Company operates an e-commerce platform and ships thousands of orders daily using third-party courier services. Since couriers charge based on order weight and delivery distance, ABC needs to verify whether the invoices match the actual expected costs. This article details the **data processing, calculations, and checks** performed to ensure fair billing.

Understanding the Datasets

ABC Company has multiple datasets that are merged and processed to verify courier charges:

1. **Website Order Report** - Contains order details, SKUs (Stock Keeping Units), and quantities.
2. **SKU Master** - Maps each SKU to its weight in grams.
3. **Pincode Mapping** - Associates warehouse and customer pincodes with delivery zones.
4. **Courier Invoice** - Contains the courier's charged weight, shipment details, and billed amount.

5. **Courier Company Rate Card** - Defines the base and additional shipping costs per weight slab and delivery zone.

Step 1: Merging the Order Report with SKU Master

Since the **Order Report** contains only SKUs and quantities, it is merged with the **SKU Master** to fetch product weights:

```
merged_order_sku = order_report.merge(sku_master, on="SKU", how="left")

merged_order_sku['Weights (Kgs)'] = merged_order_sku['Weight (g)'] / 1000
```

This converts weights from grams to kilograms for easier charge calculations.

Step 2: Weight Slab Calculation

Courier companies charge based on **rounded-up weight slabs** in increments of 0.5 kg. A function ensures that weight values conform to this billing rule:

```
def weight_slab(weight):
    i = round(weight % 1, 1)  # Extract decimal part
    if i == 0.0:
        return weight  # If it's already a whole number
    elif i > 0.5:
        return int(weight) + 1.0  # Round up to the next whole number
    else:
        return int(weight) + 0.5  # Round up to nearest 0.5 kg
```

merged_order_sku['Weight Slab (KG)'] = merged_order_sku['Weights (Kgs)'].apply(weight_slab)

Step 3: Merging Courier Invoice and Pincode Mapping

To validate the courier-assigned delivery zones, we merge the **Pincode Mapping** with the **Courier Invoice**:

abc_courier = pincode_mapping.drop_duplicates(subset=['Customer Pincode'])

courier_abc = courier_invoice[['Order ID', 'Customer Pincode', 'Type of Shipment']]

pincodes = courier_abc.merge(abc_courier, on='Customer Pincode')

This step helps verify if the courier company used the correct **zone classification** for each order.

Step 4: Calculating Expected Charges

Using the **Courier Company Rate Card**, the expected delivery charges are computed based on the zone and weight slab:

total_expected_charge = []

for _, row in merged_order_sku.iterrows():

    fwd_category = 'fwd_' + row['Delivery Zone As Per ABC']

    fwd_fixed = courier_company_rates.at[0, fwd_category + '_fixed']

    fwd_additional = courier_company_rates.at[0, fwd_category + '_additional']

    weight_slab = row['Weight Slab (KG)']

    additional_weight = max(0, (weight_slab - 0.5) / 0.5)

```
    if row['Type of Shipment'] == 'Forward charges':
        total_expected_charge.append(fwd_fixed + additional_weight * fwd_additional)
    else:
        total_expected_charge.append(0)
merged_order_sku['Expected Charge as per ABC'] = total_expected_charge
```

This logic calculates the **base cost** (fixed charge) and the **extra charge per weight slab**.

Step 5: Comparing Actual and Expected Charges

Now that we have both the **expected charges (as per ABC)** and **actual courier charges (from the invoice)**, we compare them:

```
merged_order_sku['Charge Difference'] = merged_order_sku['Expected Charge as per ABC'] - courier_invoice['Billing Amount (Rs.)']
```

If the **Charge Difference** is **positive**, ABC may have been **overcharged** by the courier. If it is **negative**, they may have been **undercharged**.

Conclusion

By following this **data-driven approach**, ABC Company can: Detect billing discrepancies. Identify incorrect weight slab calculations. Validate courier-assigned delivery zones. Ensure cost accuracy and avoid overpaying for shipments.

This automated process significantly reduces **manual effort**, enhances **financial control**, and allows ABC to dispute incorrect courier bills effectively.