SUPPLY CHAIN MANAGEMENT

Project Objective

The objective of this project is to verify the accuracy of courier charges billed to Company X by comparing the expected delivery charges (calculated using internal order data, weight slabs, delivery zones, and rate cards) with the actual charges billed by the courier company.

This project helps to:

- Identify overcharged, undercharged, and correctly charged orders
- Reduce unnecessary logistics costs
- Improve transparency and control in courier billing

About the Project

Company X ships customer orders using third-party courier services. Courier charges depend on:

- Total shipment weight
- Weight slabs (rounded up to the nearest 0.5 kg)
- Delivery zone (A, B, C, D, E)
- Type of shipment (Forward / RTO)

However, discrepancies may occur between:

- Company X's expected charges
- Courier company's billed charges

This project performs a courier charge reconciliation using Python (Jupyter Notebook) to validate billing accuracy at the order level.

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
order = pd.read_excel("Company X - Order Report.xlsx")
```

```
sku = pd.read_excel("Company X - SKU Master.xlsx")
pincode = pd.read_excel("Company X - Pincode Zones.xlsx")
invoice = pd.read_excel("Courier Company - Invoice.xlsx")
rates = pd.read_excel("Courier Company - Rates.xlsx")
```

In [3]:
```
order.head()
```

Out[3]:

	ExternOrderNo	SKU	Order Qty
0	2001827036	8904223818706	1.0
1	2001827036	8904223819093	1.0
2	2001827036	8904223819109	1.0
3	2001827036	8904223818430	1.0
4	2001827036	8904223819277	1.0

In [4]:
```
sku.head()
```

Out[4]:

	SKU	Weight (g)
0	8904223815682	210
1	8904223815859	165
2	8904223815866	113
3	8904223815873	65
4	8904223816214	120

In [5]:
```
pincode.head()
```

Out[5]:

	Warehouse Pincode	Customer Pincode	Zone
0	121003	507101	d
1	121003	486886	d
2	121003	532484	d
3	121003	143001	b
4	121003	515591	d

In [6]:
```
invoice.head()
```

Out[6]:

	AWB Code	Order ID	Charged Weight	Warehouse Pincode	Customer Pincode	Zone	Type of Shipment
0	1091117222124	2001806232	1.30	121003	507101	d	Forward charges
1	1091117222194	2001806273	1.00	121003	486886	d	Forward charges
2	1091117222931	2001806408	2.50	121003	532484	d	Forward charges
3	1091117223244	2001806458	1.00	121003	143001	b	Forward charges
4	1091117229345	2001807012	0.15	121003	515591	d	Forward charges

In [7]:
```
rates.head()
```

Out[7]:

	fwd_a_fixed	fwd_a_additional	fwd_b_fixed	fwd_b_additional	fwd_c_fixed	fwd
0	29.5	23.6	33	28.3	40.1	

In [10]:
```
print(order.columns)
```

```
Index(['ExternOrderNo', 'SKU', 'Order Qty', 'Weight (g)'], dtype='object')
```

In [11]:
```
print(sku.columns)
```

```
Index(['SKU', 'Weight (g)'], dtype='object')
```

In [12]:
```
print(pincode.columns)
```

```
Index(['Warehouse Pincode', 'Customer Pincode', 'Zone'], dtype='object')
```

In [13]:
```
print(invoice.columns)
print(rates.columns)
```

```
Index(['AWB Code', 'Order ID', 'Charged Weight', 'Warehouse Pincode',
       'Customer Pincode', 'Zone', 'Type of Shipment', 'Billing Amount (Rs.)'],
      dtype='object')
Index(['fwd_a_fixed', 'fwd_a_additional', 'fwd_b_fixed', 'fwd_b_additional',
       'fwd_c_fixed', 'fwd_c_additional', 'fwd_d_fixed', 'fwd_d_additional',
       'fwd_e_fixed', 'fwd_e_additional', 'rto_a_fixed', 'rto_a_additional',
       'rto_b_fixed', 'rto_b_additional', 'rto_c_fixed', 'rto_c_additional',
       'rto_d_fixed', 'rto_d_additional', 'rto_e_fixed', 'rto_e_additional'],
      dtype='object')
```

In [14]:
```
order_id = order["ExternOrderNo"]
sku_code = order["SKU"]
order_qty = order["Order Qty"]
order_weight_g = order["Weight (g)"]
```

In [15]:
```
sku_code_master = sku["SKU"]
sku_weight_g = sku["Weight (g)"]
```

In [16]:
```
warehouse_pincode = pincode["Warehouse Pincode"]
customer_pincode = pincode["Customer Pincode"]
delivery_zone_x = pincode["Zone"]
```

In [17]:
```
awb_code = invoice["AWB Code"]
invoice_order_id = invoice["Order ID"]
charged_weight = invoice["Charged Weight"]
invoice_warehouse_pincode = invoice["Warehouse Pincode"]
invoice_customer_pincode = invoice["Customer Pincode"]
invoice_zone = invoice["Zone"]
shipment_type = invoice["Type of Shipment"]
billed_amount = invoice["Billing Amount (Rs.)"]
```

In [18]:
```
fwd_a_fixed = rates["fwd_a_fixed"]
fwd_a_additional = rates["fwd_a_additional"]

fwd_b_fixed = rates["fwd_b_fixed"]
fwd_b_additional = rates["fwd_b_additional"]

fwd_c_fixed = rates["fwd_c_fixed"]
fwd_c_additional = rates["fwd_c_additional"]

fwd_d_fixed = rates["fwd_d_fixed"]
fwd_d_additional = rates["fwd_d_additional"]

fwd_e_fixed = rates["fwd_e_fixed"]
fwd_e_additional = rates["fwd_e_additional"]

rto_a_fixed = rates["rto_a_fixed"]
rto_a_additional = rates["rto_a_additional"]

rto_b_fixed = rates["rto_b_fixed"]
rto_b_additional = rates["rto_b_additional"]

rto_c_fixed = rates["rto_c_fixed"]
rto_c_additional = rates["rto_c_additional"]

rto_d_fixed = rates["rto_d_fixed"]
rto_d_additional = rates["rto_d_additional"]

rto_e_fixed = rates["rto_e_fixed"]
rto_e_additional = rates["rto_e_additional"]
```

In [21]:
```
total_weight_row = order_weight_g * order_qty
```

In [22]:
```
order_weight_df = pd.DataFrame({
    "ExternOrderNo": order_id,
    "total_weight_g": total_weight_row
})
```

```
order_weight_df = (
    order_weight_df
    .groupby("ExternOrderNo", as_index=False)
    .sum()
)
```

In [23]:
```
order_weight_df["weight_kg"] = order_weight_df["total_weight_g"] / 1000
order_weight_df["weight_slab_x"] = np.ceil(
    order_weight_df["weight_kg"] / 0.5
) * 0.5
```

In [24]:
```
zone_df = pd.DataFrame({
    "ExternOrderNo": order_id,
    "Warehouse Pincode": warehouse_pincode,
    "Customer Pincode": customer_pincode,
    "Zone_x": delivery_zone_x
}).drop_duplicates()
```

In [25]:
```
order_calc = order_weight_df.merge(
    zone_df[["ExternOrderNo", "Zone_x"]],
    on="ExternOrderNo",
    how="left"
)
```

In [26]:
```
invoice_df = pd.DataFrame({
    "ExternOrderNo": invoice_order_id,
    "AWB Code": awb_code,
    "Charged Weight": charged_weight,
    "Zone_y": invoice_zone,
    "Type of Shipment": shipment_type,
    "Billing Amount (Rs.)": billed_amount
})
```

In [27]:
```
final_df = order_calc.merge(
    invoice_df,
    on="ExternOrderNo",
    how="left"
)
```

In [28]:
```
def expected_charge(row):
    zone = row["Zone_x"].lower()
    slab = row["weight_slab_x"]
    ship = row["Type of Shipment"].lower()

    if ship == "forward":
        fixed = rates[f"fwd_{zone}_fixed"].iloc[0]
        add = rates[f"fwd_{zone}_additional"].iloc[0]
    else:
        fixed = rates[f"rto_{zone}_fixed"].iloc[0]
        add = rates[f"rto_{zone}_additional"].iloc[0]
```

```
    if slab <= 0.5:
        return fixed
    return fixed + ((slab - 0.5) / 0.5) * add
```

In [33]:
```
def expected_charge(row):
    zone = str(row["Zone_x"]).strip().lower()
    slab = row["weight_slab_x"]
    ship = str(row["Type of Shipment"]).strip().lower()

    if zone == "nan" or ship == "nan":
        return np.nan

    if ship == "forward":
        fixed = rates[f"fwd_{zone}_fixed"].iloc[0]
        add = rates[f"fwd_{zone}_additional"].iloc[0]
    else:
        fixed = rates[f"rto_{zone}_fixed"].iloc[0]
        add = rates[f"rto_{zone}_additional"].iloc[0]

    if slab <= 0.5:
        return fixed
    return fixed + ((slab - 0.5) / 0.5) * add
```

In [34]:
```
final_df["expected_charge_x"] = final_df.apply(expected_charge, axis=1)
```

In [35]:
```
final_df["difference"] = (
    final_df["expected_charge_x"] -
    final_df["Billing Amount (Rs.)"]
)

final_df["status"] = np.where(
    final_df["difference"] == 0,
    "Correct",
    np.where(final_df["difference"] < 0, "Overcharged", "Undercharged")
)
```

In [36]:
```
summary = final_df.groupby("status").agg(
    orders=("ExternOrderNo", "count"),
    amount=("difference", "sum")
).reset_index()

summary
```

Out[36]:

	status	orders	amount
0	Overcharged	94	-6131.6
1	Undercharged	114	980.8

In [37]:
```
final_df
```

Out[37]:

	ExternOrderNo	total_weight_g	weight_kg	weight_slab_x	Zone_x	AWI
0	2001806210	220.0	0.220	0.5	NaN	1091117
1	2001806226	480.0	0.480	0.5	NaN	1091117
2	2001806229	500.0	0.500	0.5	NaN	1091117
3	2001806232	1302.0	1.302	1.5	NaN	1091117
4	2001806233	245.0	0.245	0.5	NaN	1091117
...	...	...	...	...	...	
203	2001827036	2176.0	2.176	2.5	d	1091122
204	2001827036	2176.0	2.176	2.5	d	1091122
205	2001827036	2176.0	2.176	2.5	b	1091122
206	2001827036	2176.0	2.176	2.5	b	1091122
207	2001827036	2176.0	2.176	2.5	d	1091122

208 rows × 13 columns

In [38]:
```
final_df.head()
```

Out[38]:

	ExternOrderNo	total_weight_g	weight_kg	weight_slab_x	Zone_x	AWB C
0	2001806210	220.0	0.220	0.5	NaN	109111722
1	2001806226	480.0	0.480	0.5	NaN	109111722
2	2001806229	500.0	0.500	0.5	NaN	109111722
3	2001806232	1302.0	1.302	1.5	NaN	109111722
4	2001806233	245.0	0.245	0.5	NaN	109111722

In [39]:
```
with pd.ExcelWriter("Courier_Charges_Analysis_Final.xlsx") as writer:
    final_df.to_excel(writer, sheet_name="Order_Level_Analysis", index=False)
    summary.to_excel(writer, sheet_name="Summary", index=False)
```

Conclusion

- This project successfully validates courier billing by comparing expected delivery charges calculated using Company X’s internal data with the actual charges billed by the courier company. The analysis highlights instances of overcharging and undercharging, helping the company improve logistics cost management and ensure billing accuracy.

Final Output

- The final output of this project is generated in an Excel file containing both the order-level analysis and the summary of overcharged, undercharged, and correctly charged orders.

Output File Name:

- Courier_Charges_Analysis_Final.xlsx

In [ ]:
```
```