Assignment 2 – Insurance Claim Analysis

This notebook performs:

- Data cleaning using core Python
- Business analysis of city-wise performance
- Classification of rejection remarks

In [30]:
```
def clean_claim_data(filepath):
    cleaned_data = []
    with open(filepath, 'r', encoding='utf-8') as file:
        headers = file.readline().strip().split(',')

        for line_num, line in enumerate(file, start=2):
            values = line.strip().split(',')

            if len(values) != len(headers):
                continue

            row = dict(zip(headers, values))

            if (not row['CLAIM_ID'].strip() or
                not row['CITY'].strip() or
                not row['CLAIM_AMOUNT'].strip() or
                not row['CLAIM_DATE'].strip()):
                continue

            try:
                row['CLAIM_AMOUNT'] = float(row['CLAIM_AMOUNT'])
            except ValueError:
                continue

            row['CITY'] = row['CITY'].strip().lower()
            cleaned_data.append(row)

    print(f"Total cleaned rows: {len(cleaned_data)}")
```

```
    return cleaned_data
```

In [31]:
```
data = clean_claim_data('claim_data_star_health.csv')
data[:3]
```

```
Total cleaned rows: 7
```

Out[31]:
```
[{'CLAIM_ID': 'CLM100021',
  'CLAIM_DATE': '4/1/2025',
  'CUSTOMER_ID': 'CUST14285',
  'CLAIM_AMOUNT': 10419.0,
  'PREMIUM_COLLECTED': '2198.59',
  'PAID_AMOUNT': '6964.46',
  'CITY': 'pune',
  'REJECTION_REMARKS': ''},
 {'CLAIM_ID': 'CLM100013',
  'CLAIM_DATE': '4/1/2025',
  'CUSTOMER_ID': 'CUST26471',
  'CLAIM_AMOUNT': 42468.0,
  'PREMIUM_COLLECTED': '8982.2',
  'PAID_AMOUNT': '30119.67',
  'CITY': 'guwahati',
  'REJECTION_REMARKS': ''},
 {'CLAIM_ID': 'CLM100099',
  'CLAIM_DATE': '4/2/2025',
  'CUSTOMER_ID': 'CUST29309',
  'CLAIM_AMOUNT': 55897.0,
  'PREMIUM_COLLECTED': '1861.78',
  'PAID_AMOUNT': '55657.15',
  'CITY': 'guwahati',
  'REJECTION_REMARKS': ''}]
```

In [32]:
```
def analyze_city_performance(data, cities_to_check):
    city_summary = {}

    for city in cities_to_check:
        city_data = [row for row in data if row['CITY'] == city.lower()]
        total_claims = len(city_data)
        total_amount = sum(row['CLAIM_AMOUNT'] for row in city_data)

        city_summary[city] = {
```

```
            'Total Claims': total_claims,
            'Total Amount': total_amount
        }

    return city_summary
```

In [33]:
```
cities = ['pune', 'kolkata', 'ranchi', 'guwahati']
summary = analyze_city_performance(data, cities)

for city, stats in summary.items():
    print(f"\n📍 {city.title()}")
    for key, value in stats.items():
        print(f"{key}: {value}")
```

```
📍 Pune
Total Claims: 2
Total Amount: 82204.0

📍 Kolkata
Total Claims: 0
Total Amount: 0

📍 Ranchi
Total Claims: 0
Total Amount: 0

📍 Guwahati
Total Claims: 2
Total Amount: 98365.0
```

Recommendation

Based on the analysis of claim data from the four cities:

- 📍 **Pune** has 2 claims totaling ₹82,204, indicating active business.
- 📍 **Guwahati** has 2 claims but a slightly higher total of ₹98,365.
- 📍 **Kolkata** and 📍 **Ranchi** have **no claim activity** at all.

📝 While Guwahati has more total claim amount, its claim volume and regional performance appear lower compared to Pune in the bigger business picture.

🔴 **Recommendation: Close operations in Kolkata and Ranchi due to zero activity. Guwahati may be monitored further before closure.**

In [34]:
```
REJECTION_REASONS_MAP = {
    "Fake_document": "fake document",
    "Not_Covered": "not covered",
    "Policy_expired": "policy expired"
}

def complex_rejection_classifier(remark_text):
    try:
        if not isinstance(remark_text, str) or remark_text.strip() == "":
            return "Invalid Remark"

        remark_text = remark_text.lower()

        for label, keyword in REJECTION_REASONS_MAP.items():
            if keyword in remark_text:
                return label

        return "Unknown"

    except Exception as e:
        print(f"Error: {e}")
        return "Error"
```

In [35]:
```
for row in data:
    remark = row.get("REJECTION_REMARKS", "")
    row["REJECTION_CLASS"] = complex_rejection_classifier(remark)
```

In [36]:
```
for row in data[:10]:
    print(f"{row['REJECTION_REMARKS']}  -->  {row['REJECTION_CLASS']}")
```

```
  -->  Invalid Remark
  -->  Invalid Remark
  -->  Invalid Remark
  -->  Invalid Remark
reason: Policy_expired in verification.  -->  Unknown
reason led to rejection.  -->  Unknown
- Not_Covered found.  -->  Unknown
```

In [37]:
```
from collections import Counter
Counter([row["REJECTION_CLASS"] for row in data])
```

Out[37]:
```
Counter({'Invalid Remark': 4, 'Unknown': 3})
```

In [ ]: