Insurance Workflow Explanation

1. Overview

The workflow automates insurance-related tasks using n8n. It handles:

1. High claim alerts (₹5 lakh+)
2. Fraud detection alerts (fraud score > 80)
3. Policy renewal reminders (30 days before expiry)
4. Logging claim data into Google Sheets
5. Optional: Telegram group notifications

The workflow is triggered by incoming claim data via a webhook.

---

2. Workflow Architecture

- Webhook Node: Receives claim data (claim_id, claim_amount, fraud_score, policy_expiry).
- IF Node(s): Checks conditions for high claims, fraud score, or renewal reminders.
- Gmail Node(s): Sends email alerts to relevant stakeholders.
- Google Sheet Node: Logs every claim with timestamp for record-keeping.
- Telegram Node (Optional): Sends group alerts for urgent cases.

---

3. Node Details

3.1 Webhook Node

- Receives JSON data from Python script or other sources.
- Temporary URL for testing: /webhook-test/insurance-alert
- Production URL (after activating workflow): /webhook/insurance-alert
- Example JSON payload:

```
{
  "claim_id": "C12345",
  "claim_amount": 650000,
  "fraud_score": 72,
  "policy_expiry": "2025-11-06"
}
```

3.2 IF Nodes

Purpose: Evaluate conditions to decide the workflow path.

1. High Claim Check
    - Condition: claim_amount > 500000
    - True → Gmail Alert to insurance team
2. Fraud Check
    - Condition: fraud_score > 80
    - True → Gmail Alert to underwriter
3. Policy Renewal Check (Optional)
    - Condition: policy_expiry < 30 days
    - True → Gmail Reminder to customer

3.3 Gmail Node

- Sends email alerts based on IF node evaluation.
- Requires OAuth connection to Gmail account.
- Alert examples:
    - High Claim Alert: “Claim C12345 exceeds ₹5 lakh.”
    - Fraud Alert: “Claim C12345 has fraud score 85, please review.”
    - Renewal Reminder: “Policy P56789 expires in 30 days.”

3.4 Google Sheet Node

- Appends every claim to a Google Sheet for record-keeping.

- Columns:

```
Claim ID | Claim Amount | Fraud Score | Policy Expiry | Timestamp
```

- Provides a permanent log of all processed claims.

3.5 Telegram Node (Optional)

- Sends group alerts for urgent claims.
- Free and effective for team notifications.
- Requires:
    - Telegram Bot Token (from @BotFather)
    - Chat ID of the group

---

4. Python Test Script

- Automates sending 20 sample claims to the webhook.
- Configurable for development or production URL.
- Prints status of each claim submission.

5. Workflow Execution

1. Python script sends JSON → Webhook receives it.
2. IF nodes evaluate conditions: High Claim / Fraud / Renewal.
3. Gmail nodes send alerts to respective stakeholders.
4. Google Sheet node logs all claims automatically.
5. Optional Telegram node sends urgent alerts.

All steps are automated; the user only needs to send claim JSON.

---

6. Notes & Best Practices

- Test with Dev Webhook first to avoid sending incorrect data.
- Activate workflow to get permanent production URL.
- Keep Google Sheet and Gmail credentials secured.
- Use Telegram only for team notifications, not sensitive data.

- **Add timestamps to track when claims are processed.**

```python
import requests

webhook_url = "http://localhost:5678/webhook/insurance-alert"

claims = [
    {"claim_id": f"C{10000+i}", "claim_amount": 500000+i*20000, "fraud_score": 50+i, "policy_expiry": "2025-11-06"}
    for i in range(20)
]

for claim in claims:
    r = requests.post(webhook_url, json=claim)
    print(f"Sent {claim['claim_id']}, status: {r.status_code}")
```