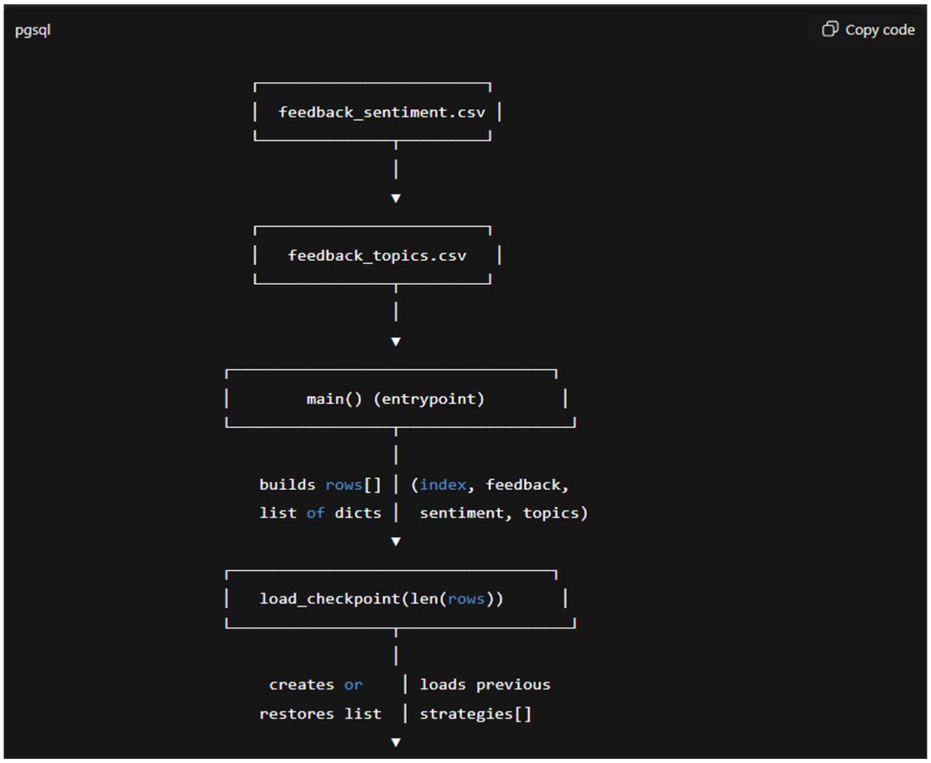
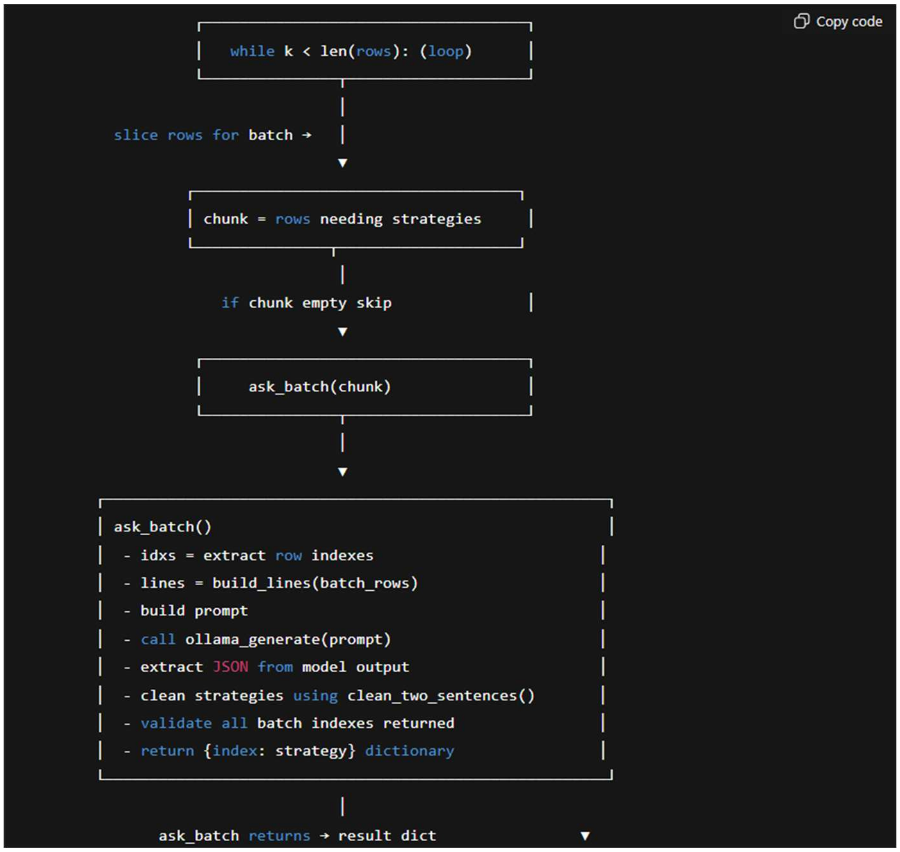
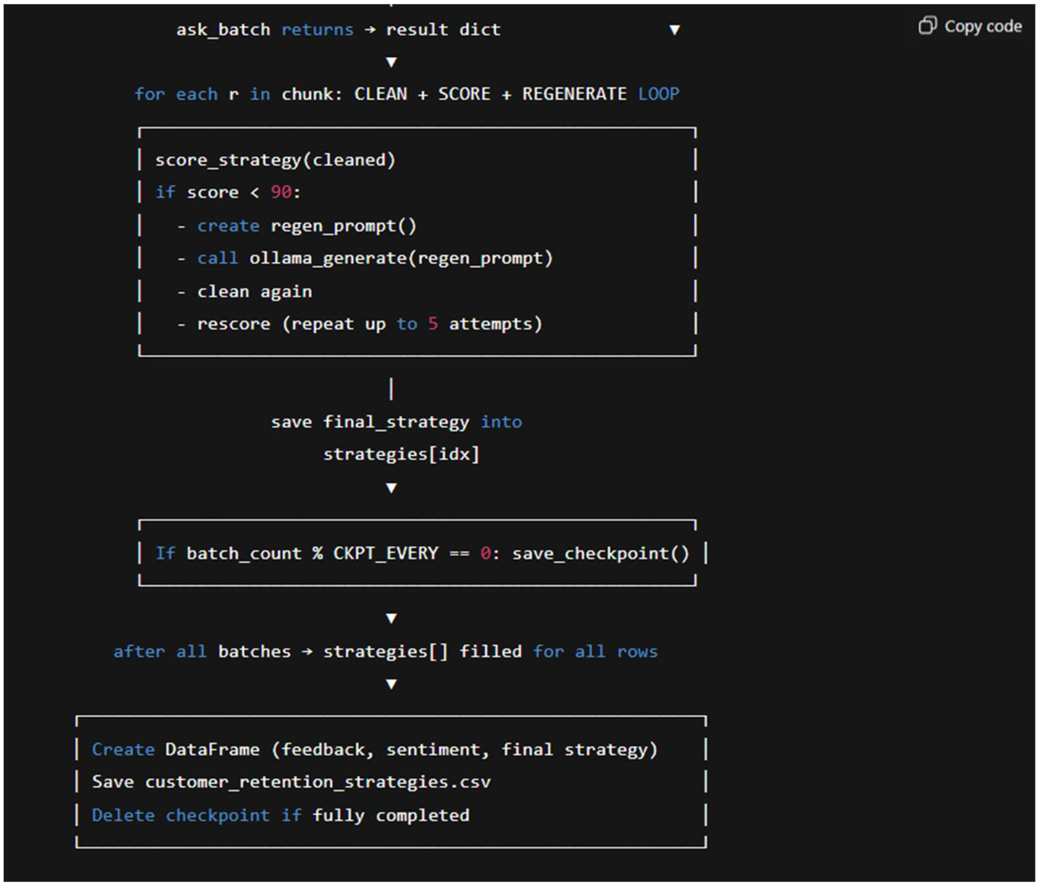
Strategy_Generator.py----High Level Workflow



```
                 ┌──────────────────────────┐
                 │  feedback_sentiment.csv  │
                 └────────────┬─────────────┘
                              │
                              ▼
                 ┌──────────────────────────┐
                 │   feedback_topics.csv    │
                 └────────────┬─────────────┘
                              │
                              ▼
              ┌────────────────────────────────────┐
              │        main()  (entrypoint)        │
              └───────────────┬────────────────────┘
                              │
              builds rows[]   │  (index, feedback,
              list of dicts   │   sentiment, topics)
                              ▼
              ┌────────────────────────────────────┐
              │   load_checkpoint(len(rows))       │
              └───────────────┬────────────────────┘
                              │
                creates or    │ loads previous
              restores list   │ strategies[]
                              ▼
```

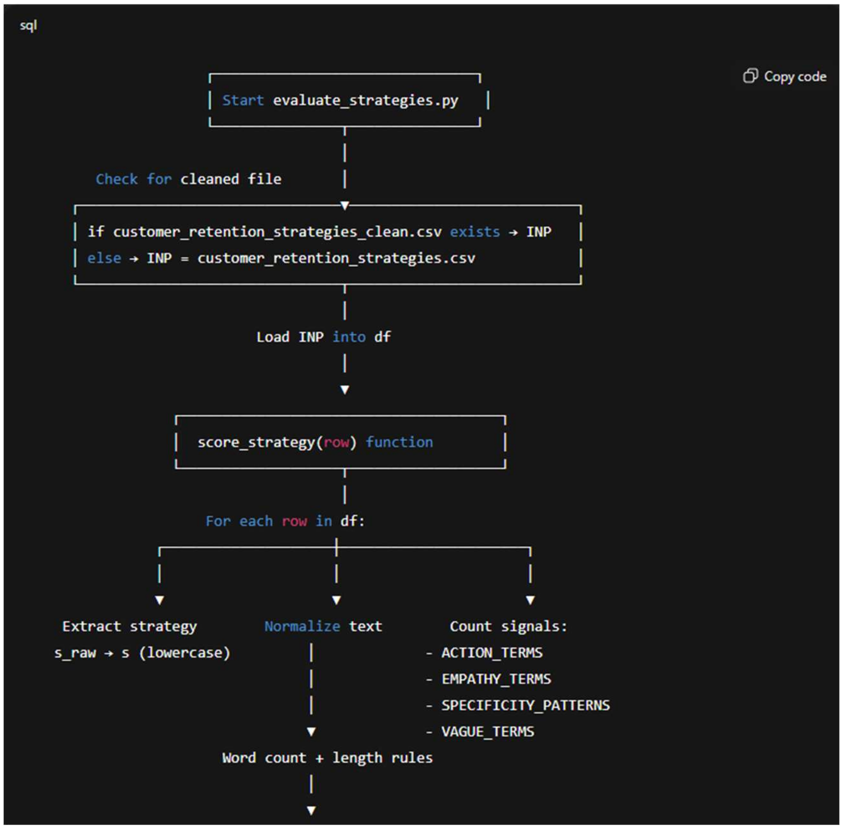


```
              ┌────────────────────────────────────┐
              │   while k < len(rows): (loop)      │
              └───────────────┬────────────────────┘
                              │
     slice rows for batch →   │
                              ▼
             ┌────────────────────────────────────┐
             │ chunk = rows needing strategies    │
             └───────────────┬────────────────────┘
                              │
            if chunk empty skip                   │
                              ▼
              ┌────────────────────────────────────┐
              │     ask_batch(chunk)               │
              └───────────────┬────────────────────┘
                              │
                              ▼
   ┌──────────────────────────────────────────────────────────┐
   │ ask_batch()                                              │
   │  - idxs = extract row indexes                            │
   │  - lines = build_lines(batch_rows)                       │
   │  - build prompt                                          │
   │  - call ollama_generate(prompt)                          │
   │  - extract JSON from model output                        │
   │  - clean strategies using clean_two_sentences()          │
   │  - validate all batch indexes returned                   │
   │  - return {index: strategy} dictionary                   │
   └──────────────────────────────────────────────────────────┘
                              │
         ask_batch returns → result dict                 ▼
```

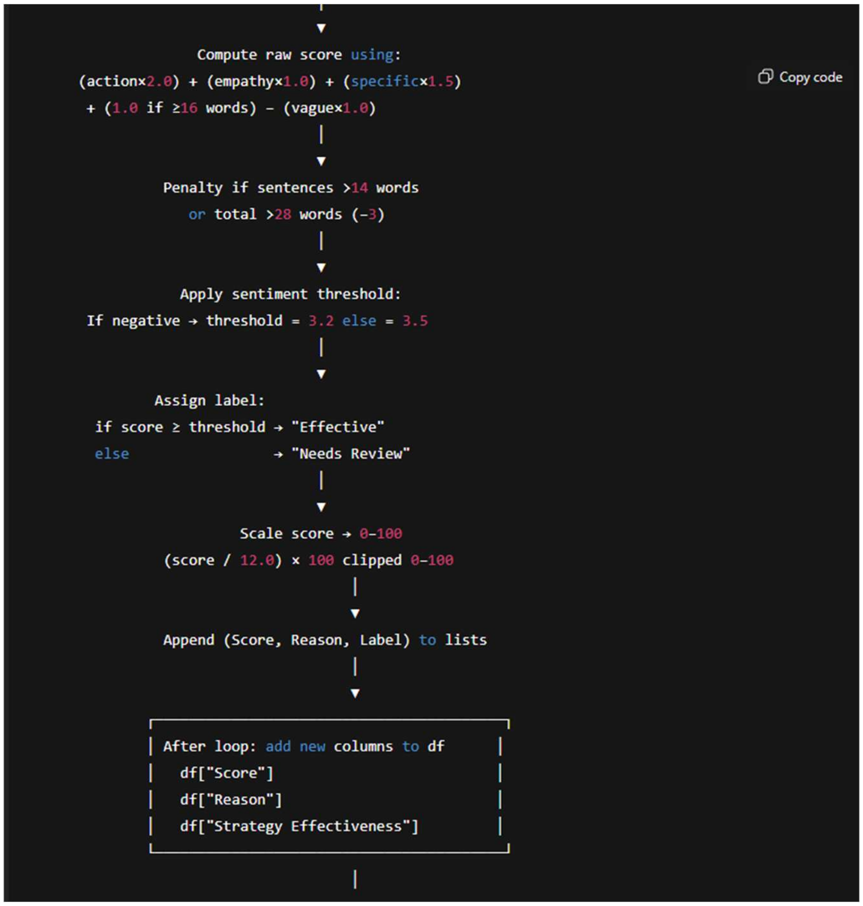


```
                              |
          ask_batch returns → result dict                        ▼
                              ▼
      for each r in chunk: CLEAN + SCORE + REGENERATE LOOP
      ┌───────────────────────────────────────────────────┐
      │ score_strategy(cleaned)                            │
      │ if score < 90:                                     │
      │   - create regen_prompt()                          │
      │   - call ollama_generate(regen_prompt)             │
      │   - clean again                                    │
      │   - rescore (repeat up to 5 attempts)              │
      └───────────────────────────────────────────────────┘
                              |
                  save final_strategy into
                       strategies[idx]
                              ▼
      ┌───────────────────────────────────────────────────┐
      │ If batch_count % CKPT_EVERY == 0: save_checkpoint() │
      └───────────────────────────────────────────────────┘
                              ▼
    after all batches → strategies[] filled for all rows
                              ▼
┌────────────────────────────────────────────────────────────┐
│ Create DataFrame (feedback, sentiment, final strategy)     │
│ Save customer_retention_strategies.csv                     │
│ Delete checkpoint if fully completed                       │
└────────────────────────────────────────────────────────────┘
```

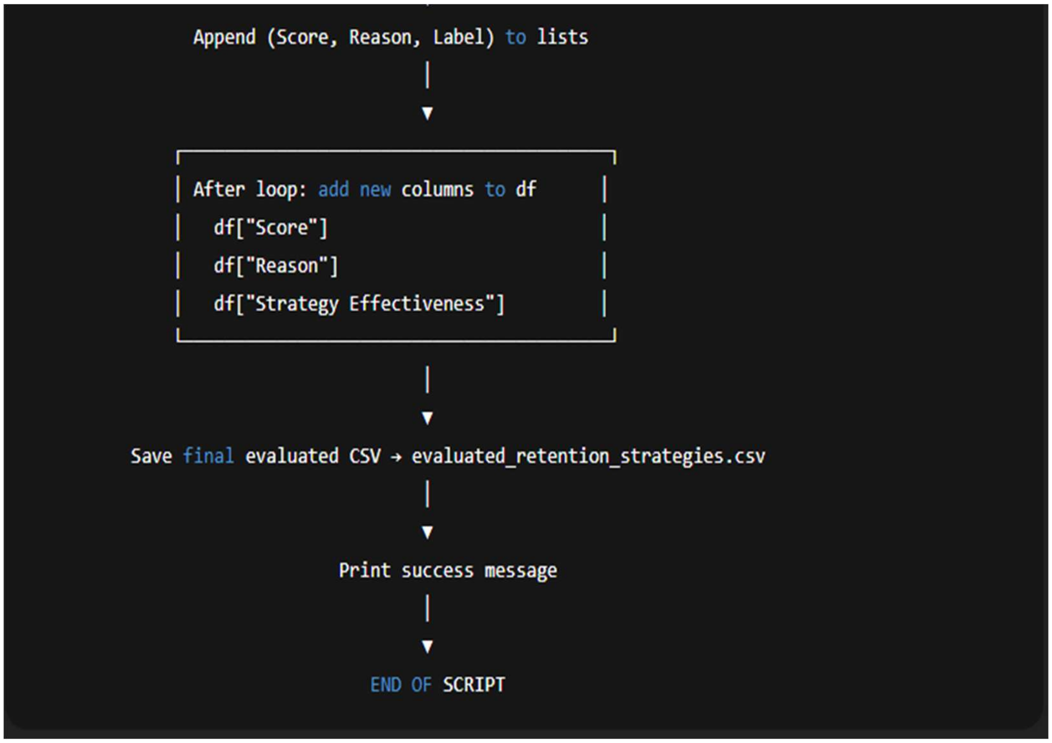
Evaluate_Strategy.py High Level Workflow



```
                    ┌────────────────────────────────┐
                    │ Start evaluate_strategies.py   │
                    └───────────────┬────────────────┘
                                    │
          Check for cleaned file    │
          ┌─────────────────────────▼──────────────────────────────┐
          │ if customer_retention_strategies_clean.csv exists → INP │
          │ else → INP = customer_retention_strategies.csv          │
          └─────────────────────────┬──────────────────────────────┘
                                    │
                          Load INP into df
                                    │
                                    ▼
                ┌──────────────────────────────────────┐
                │  score_strategy(row) function         │
                └───────────────────┬──────────────────┘
                                    │
                           For each row in df:
               ┌────────────────────┼───────────────────────┐
               │                    │                       │
               ▼                    ▼                       ▼
        Extract strategy      Normalize text          Count signals:
      s_raw → s (lowercase)       │                 - ACTION_TERMS
                                  │                 - EMPATHY_TERMS
                                  │                 - SPECIFICITY_PATTERNS
                                  ▼                 - VAGUE_TERMS
                      Word count + length rules
                                  │
                                  ▼
```

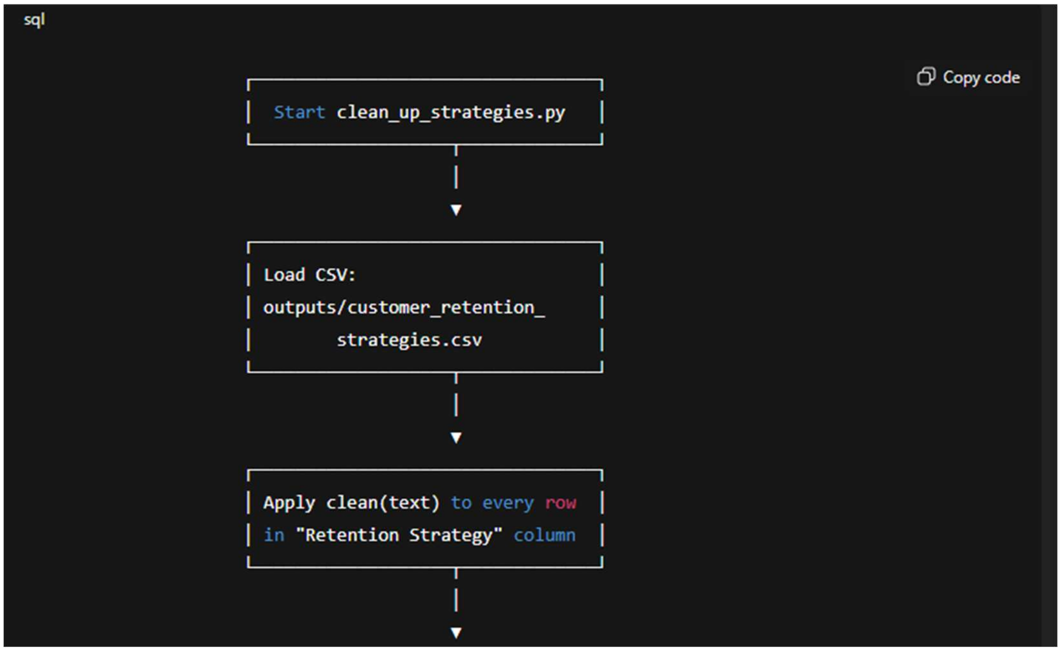


```
                                  ▼
                      Compute raw score using:
          (action×2.0) + (empathy×1.0) + (specific×1.5)
           + (1.0 if ≥16 words) − (vague×1.0)
                                  │
                                  ▼
                  Penalty if sentences >14 words
                     or total >28 words (−3)
                                  │
                                  ▼
                    Apply sentiment threshold:
          If negative → threshold = 3.2 else = 3.5
                                  │
                                  ▼
                 Assign label:
           if score ≥ threshold → "Effective"
           else                 → "Needs Review"
                                  │
                                  ▼
                        Scale score → 0–100
                  (score / 12.0) × 100 clipped 0–100
                                      │
                                      ▼
                  Append (Score, Reason, Label) to lists
                                      │
                                      ▼
                 ┌─────────────────────────────────────────┐
                 │ After loop: add new columns to df        │
                 │   df["Score"]                            │
                 │   df["Reason"]                           │
                 │   df["Strategy Effectiveness"]           │
                 └─────────────────────────────────────────┘
                                      │
```

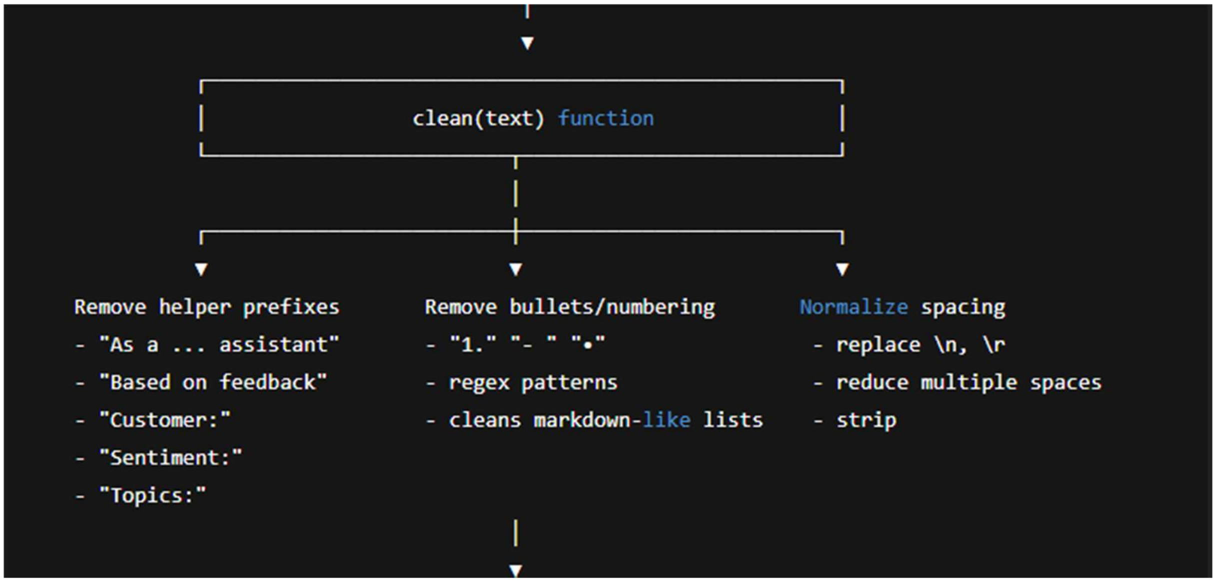


```
                Append (Score, Reason, Label) to lists
                                |
                                ▼
              ┌─────────────────────────────────────┐
              │ After loop: add new columns to df    │
              │   df["Score"]                        │
              │   df["Reason"]                       │
              │   df["Strategy Effectiveness"]       │
              └─────────────────────────────────────┘
                                |
                                ▼
       Save final evaluated CSV → evaluated_retention_strategies.csv
                                |
                                ▼
                        Print success message
                                |
                                ▼
                           END OF SCRIPT
```

Clean_Strategy.py High Level Workflow

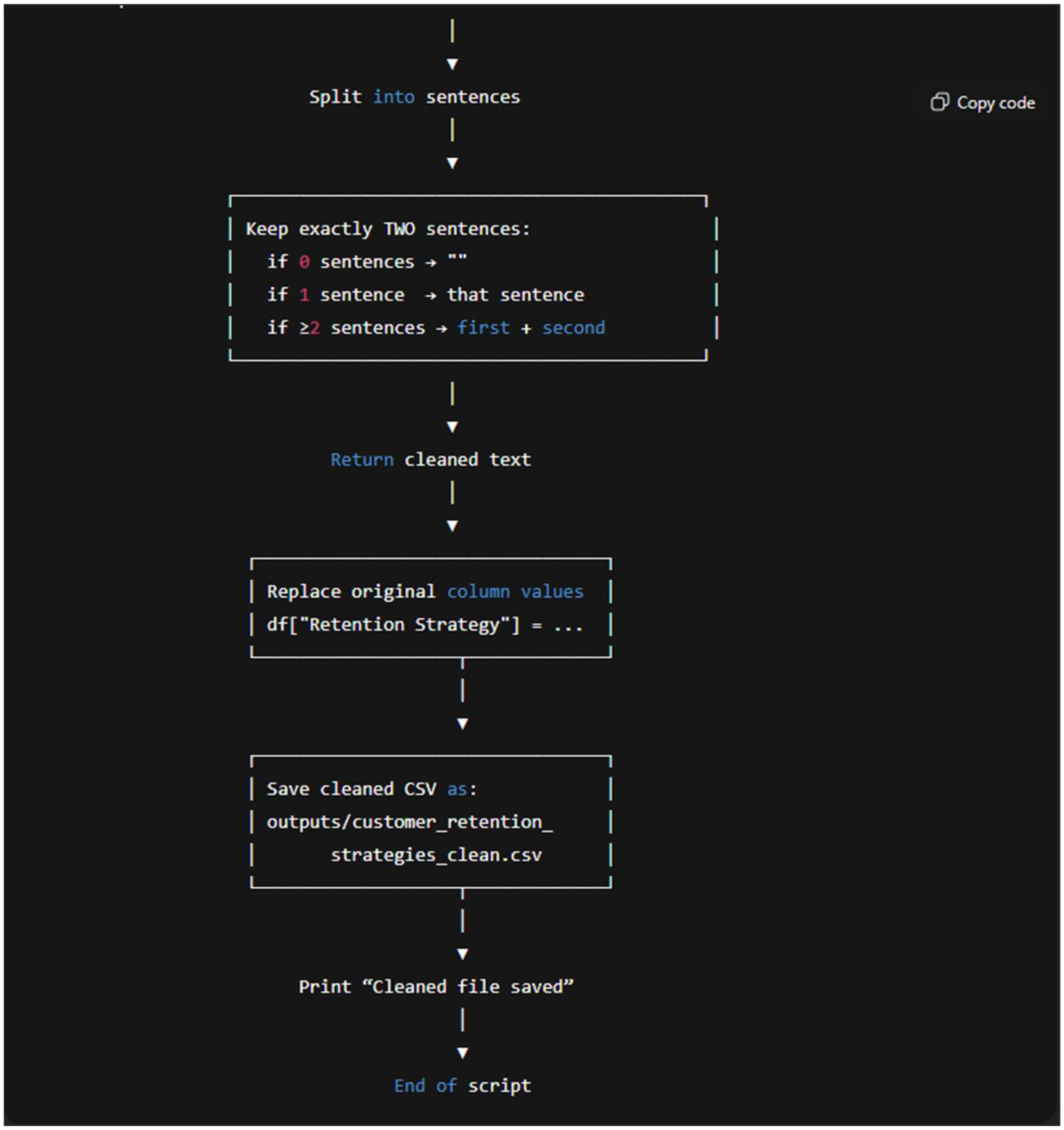


```
┌──────────────────────────────┐
│  Start clean_up_strategies.py │
└──────────────────────────────┘
               │
               │
               ▼
┌──────────────────────────────┐
│ Load CSV:                     │
│ outputs/customer_retention_   │
│       strategies.csv          │
└──────────────────────────────┘
               │
               │
               ▼
┌──────────────────────────────┐
│ Apply clean(text) to every row│
│ in "Retention Strategy" column│
└──────────────────────────────┘
               │
               │
               ▼
```



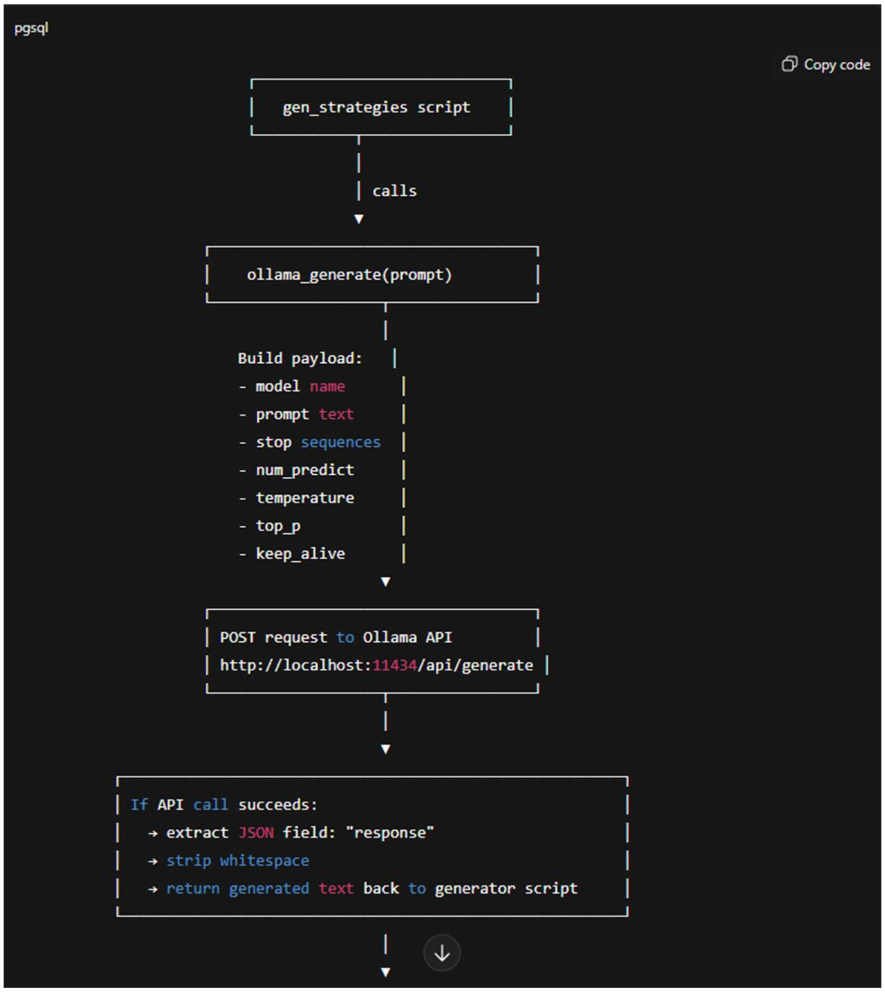
```
                    │
                    ▼
          ┌────────────────────────────────────┐
          │           clean(text) function      │
          └────────────────────────────────────┘
                           │
                           │
          ┌────────────────┼────────────────┐
          ▼                ▼                ▼
Remove helper prefixes    Remove bullets/numbering    Normalize spacing
- "As a ... assistant"    - "1." "- " "•"              - replace \n, \r
- "Based on feedback"     - regex patterns             - reduce multiple spaces
- "Customer:"             - cleans markdown-like lists - strip
- "Sentiment:"
- "Topics:"

                           │
                           ▼
```

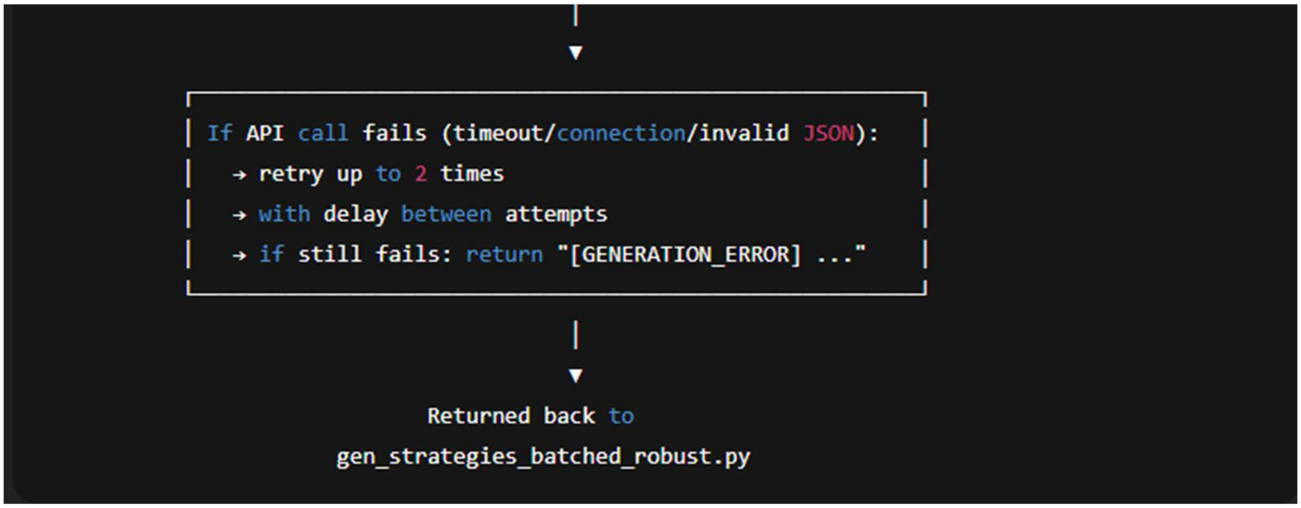


```
                     │
                     ▼
           Split into sentences
                     │
                     ▼
       ┌──────────────────────────────────┐
       │ Keep exactly TWO sentences:      │
       │   if 0 sentences → ""            │
       │   if 1 sentence  → that sentence │
       │   if ≥2 sentences → first + second │
       └──────────────────────────────────┘
                     │
                     ▼
             Return cleaned text
                     │
                     ▼
         ┌─────────────────────────┐
         │ Replace original column values │
         │ df["Retention Strategy"] = ... │
         └─────────────────────────┘
                      │
                      ▼
         ┌─────────────────────────┐
         │ Save cleaned CSV as:    │
         │ outputs/customer_retention_ │
         │       strategies_clean.csv │
         └─────────────────────────┘
                      │
                      ▼
          Print “Cleaned file saved”
                      │
                      ▼
                End of script
```

Ollama_Utils.py High level workflow



```
                 ┌──────────────────────┐
                 │  gen_strategies script │
                 └──────────────────────┘
                            │
                            │ calls
                            ▼
              ┌──────────────────────────┐
              │    ollama_generate(prompt)   │
              └──────────────────────────┘
                               │
                Build payload:   │
                - model name      │
                - prompt text     │
                - stop sequences  │
                - num_predict     │
                - temperature     │
                - top_p           │
                - keep_alive      │
                               ▼
              ┌──────────────────────────┐
              │ POST request to Ollama API  │
              │ http://localhost:11434/api/generate │
              └──────────────────────────┘
                               │
                               ▼
      ┌────────────────────────────────────────┐
      │ If API call succeeds:                       │
      │   → extract JSON field: "response"          │
      │   → strip whitespace                        │
      │   → return generated text back to generator script │
      └────────────────────────────────────────┘
                               │
                               ▼
```



```
                               │
                               ▼
      ┌────────────────────────────────────────┐
      │ If API call fails (timeout/connection/invalid JSON): │
      │   → retry up to 2 times                     │
      │   → with delay between attempts             │
      │   → if still fails: return "[GENERATION_ERROR] ..." │
      └────────────────────────────────────────┘
                               │
                               ▼
                     Returned back to
               gen_strategies_batched_robust.py
```