Prompt:1:

def llm\_complex\_clean(record):

prompt = f"""

Given the following record:

1. \*\*Fill Missing Values\*\*

- If 'city' is missing or null, infer it from 'incident\_address' when possible.

2. \*\*Standardize Text\*\*

- Trim whitespace.

- Use Title Case for categorical fields (e.g. 'complaint\_type', 'descriptor').

- Correct common misspellings.

3. \*\*Validate & Format\*\*

- 'incident\_address' → format as "Street, Borough, NY" (e.g. "123 Broadway, Manhattan, NY").

- 'created\_date' → ISO 8601 (YYYY‑MM‑DDTHH:MM:SS).

- Ensure 'open\_data\_channel\_type' is a known channel (e.g. "Call", "Web", "Mobile App").

4. \*\*Categorize\*\*

- Add a new key 'complaint\_category':

- If descriptor or complaint\_type contains "Noise", set to "Noise".

- Otherwise set to "Non-Noise".

Return \*\*only\*\* a Python dict literal with all original keys (cleaned) plus 'complaint\_category'.

{record.to\_dict()}

"""

Prompt2:

prompt = f"""

You are a Python data‐cleaning and validation specialist and code generator.

Given the cleaned NYC 311 record as a Python dict:

{record}

Your tasks:

1. Output the cleaned record as a valid Python dict literal.

2. Write standalone Python code using only `assert` statements (no `unittest` framework) to verify:

- \*\*Data types\*\*: each field has the correct type (`int`, `str`, `float`, or `None`).

- \*\*Value ranges\*\*: e.g. `latitude` ∈ [-90, 90], `longitude` ∈ [-180, 180].

- \*\*Non‐null requirements\*\*: fields `unique\_key`, `created\_date`, `complaint\_type`, `borough`, and `incident\_zip` are not `None` or empty.

- \*\*Categorical checks\*\*: `status` ∈ {'Open', 'Closed', 'In Progress'}.

3. After the assertions, include `print()` statements indicating success.

\*\*Output format\*\* (in a Colab notebook):

```python

# 1) Cleaned record dict

cleaned = { ... }

# 2) Validation asserts

assert ...

...

# 3) Success messages

print("All checks passed")

"""