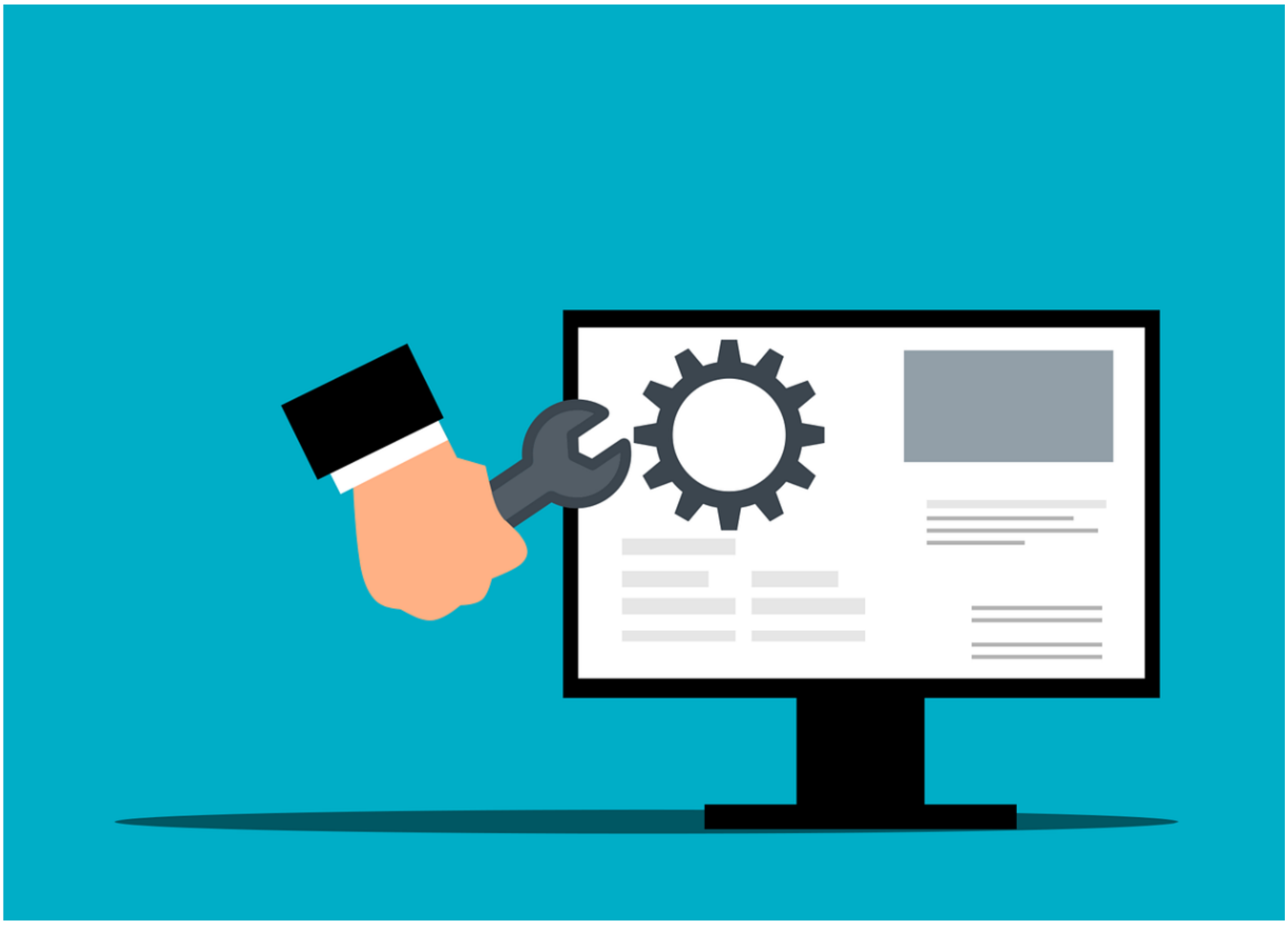
Description



Business Context

In today's dynamic business landscape, organizations are increasingly recognizing the pivotal role customer feedback plays in shaping the trajectory of their products and services. The ability to swiftly and **effectively respond to customer** input not only fosters enhanced customer experiences but also serves as a catalyst for growth, prolonged customer engagement, and the nurturing of lifetime value relationships. As a dedicated Product Manager or Product Analyst, staying attuned to the voice of your customers is not just a best practice; it's a strategic imperative. While your organization may be inundated with a wealth of customer-generated feedback and support tickets, your role entails much more than just processing these inputs. To make your efforts in managing customer experience and expectations truly impactful, you need a structured approach – a method that allows you to discern the most pressing issues, set priorities, and allocate resources judiciously. One of the most effective strategies at your disposal as an organization is to harness the power of automated Support Ticket Categorization - **done in the modern day using Large Language Models and Generative AI.**

Objective

Develop a Generative AI application using a Large Language Model to automate the classification and processing of support tickets. **The application will aim to predict ticket categories, assign priority, suggest estimated resolution times, and store the results in a structured DataFrame.**

Installing and Importing Necessary Libraries

In [ ]:

```
# Installation for GPU llama-cpp-python
!CMAKE_ARGS="-DLLAMA_CUBLAS=on" FORCE_CMAKE=1 pip install llama-cpp-python --force-reins
tall --upgrade --no-cache-dir -q
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.5/42.5 MB 57.4 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.2/18.2 MB 53.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.5/45.5 kB 189.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.2/133.2 kB 300.3 MB/s eta 0:00:00
  Building wheel for llama-cpp-python (pyproject.toml) ... done
ERROR: pip's dependency resolver does not currently take into account all the packages th
at are installed. This behaviour is the source of the following dependency conflicts.
torch 2.2.1+cu121 requires nvidia-cublas-cu12==12.1.3.1; platform_system == "Linux" and p
latform_machine == "x86_64", which is not installed.
torch 2.2.1+cu121 requires nvidia-cuda-cupti-cu12==12.1.105; platform_system == "Linux" a
nd platform_machine == "x86_64", which is not installed.
torch 2.2.1+cu121 requires nvidia-cuda-nvrtc-cu12==12.1.105; platform_system == "Linux" a
nd platform_machine == "x86_64", which is not installed.
torch 2.2.1+cu121 requires nvidia-cuda-runtime-cu12==12.1.105; platform_system == "Linux"
and platform_machine == "x86_64", which is not installed.
torch 2.2.1+cu121 requires nvidia-cudnn-cu12==8.9.2.26; platform_system == "Linux" and pl
atform_machine == "x86_64", which is not installed.
torch 2.2.1+cu121 requires nvidia-cufft-cu12==11.0.2.54; platform_system == "Linux" and p
latform_machine == "x86_64", which is not installed.
torch 2.2.1+cu121 requires nvidia-curand-cu12==10.3.2.106; platform_system == "Linux" and
platform_machine == "x86_64", which is not installed.
torch 2.2.1+cu121 requires nvidia-cusolver-cu12==11.4.5.107; platform_system == "Linux" a
nd platform_machine == "x86_64", which is not installed.
torch 2.2.1+cu121 requires nvidia-cusparse-cu12==12.1.0.106; platform_system == "Linux" a
nd platform_machine == "x86_64", which is not installed.
torch 2.2.1+cu121 requires nvidia-nccl-cu12==2.19.3; platform_system == "Linux" and platf
orm_machine == "x86_64", which is not installed.
torch 2.2.1+cu121 requires nvidia-nvtx-cu12==12.1.105; platform_system == "Linux" and pla
tform_machine == "x86_64", which is not installed.
```

In [ ]:

```
# Install the hugging face hub
!pip install huggingface_hub -q
```

In [ ]:

```
# Importing library for data manipulation
import pandas as pd

# Function to download the model from the Hugging Face model hub
from huggingface_hub import hf_hub_download

# Importing the Llama class from the llama_cpp module
from llama_cpp import Llama

# Importing the json module
import json
```

1. Data Overview:

Load the dataset - Print the overview of the data (first few rows, shape, etc)

In [ ]:

```
# Mount Google Drive
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

In [ ]:

```
# Load the dataset
file_path="/content/drive/MyDrive/Support_ticket_text_data_mid_term.csv"
data = pd.read_csv(file_path)
```

In [ ]:

```
# Print the first few rows
print("First few rows of the dataset:")
data.head()
```

```
First few rows of the dataset:
```

Out[ ]:

	support_tick_id	support_ticket_text
0	ST2023-006	My internet connection has significantly slowe...
1	ST2023-007	Urgent help required! My laptop refuses to sta...
2	ST2023-008	I've accidentally deleted essential work docum...
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...
4	ST2023-010	My smartphone battery is draining rapidly, eve...

In [ ]:

```
# Print the shape of the DataFrame
print("\nShape of the dataset:")
print(data.shape)
```

```
Shape of the dataset:
(21, 2)
```

- **We have 21 rows and 2 columns in the data.**

In [ ]:

```
# Check for missing values
missing_values= data.isnull().sum()

if missing_values.any():
    print("There are missing values.")
else:
    print("There are no missing values.")
```

```
There are no missing values.
```

Model building

Load the model from Hugging Face - Create a function to define the model parameters and generate a response

In [ ]:

```
# Loading the model
model_name_or_path = "TheBloke/Llama-2-13B-chat-GGUF"
model_basename = "llama-2-13b-chat.Q5_K_M.gguf"
```

In [ ]:

```
# Download the model from the Hugging Face Hub using the 'hf_hub_download' function

model_path = hf_hub_download(
    repo_id=model_name_or_path,
    filename=model_basename
    )
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
```

```
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://h
uggingface.co/settings/tokens), set it as secret in your Google Colab and restart your se
ssion.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models
or datasets.
  warnings.warn(
```

In [ ]:

```
# Create an instance of the 'Llama' class with specified parameters
lcpp_llm = Llama(
        model_path=model_path,
        n_threads=2,  # CPU cores
        n_batch=512,  # Should be between 1 and n_ctx, consider the amount of VRAM in you
r GPU.
        n_gpu_layers=43,  # Change this value based on your model and your GPU VRAM pool
.
        n_ctx=4096,  # Context window
    )
```

```
llama_model_loader: loaded meta data with 19 key-value pairs and 363 tensors from /root/.
cache/huggingface/hub/models--TheBloke--Llama-2-13B-chat-GGUF/snapshots/4458acc949de0a991
4c3eab623904d4fe999050a/llama-2-13b-chat.Q5_K_M.gguf (version GGUF V2)
llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not apply in this
output.
llama_model_loader: - kv   0:                       general.architecture str
= llama
llama_model_loader: - kv   1:                               general.name str
= LLaMA v2
llama_model_loader: - kv   2:                       llama.context_length u32
= 4096
llama_model_loader: - kv   3:                     llama.embedding_length u32
= 5120
llama_model_loader: - kv   4:                          llama.block_count u32
= 40
llama_model_loader: - kv   5:                  llama.feed_forward_length u32
= 13824
llama_model_loader: - kv   6:                 llama.rope.dimension_count u32
= 128
llama_model_loader: - kv   7:                 llama.attention.head_count u32
= 40
llama_model_loader: - kv   8:              llama.attention.head_count_kv u32
= 40
llama_model_loader: - kv   9:     llama.attention.layer_norm_rms_epsilon f32
= 0.000010
llama_model_loader: - kv  10:                          general.file_type u32
= 17
llama_model_loader: - kv  11:                       tokenizer.ggml.model str
= llama
llama_model_loader: - kv  12:                      tokenizer.ggml.tokens arr[str,32000]
= ["<unk>", "<s>", "</s>", "<0x00>", "<...
llama_model_loader: - kv  13:                      tokenizer.ggml.scores arr[f32,32000]
= [0.000000, 0.000000, 0.000000, 0.0000...
llama_model_loader: - kv  14:                  tokenizer.ggml.token_type arr[i32,32000]
= [2, 3, 3, 6, 6, 6, 6, 6, 6, 6, 6, 6, ...
llama_model_loader: - kv  15:                tokenizer.ggml.bos_token_id u32
= 1
llama_model_loader: - kv  16:                tokenizer.ggml.eos_token_id u32
= 2
llama_model_loader: - kv  17:            tokenizer.ggml.unknown_token_id u32
= 0
llama_model_loader: - kv  18:               general.quantization_version u32
= 2
llama_model_loader: - type  f32:   81 tensors
llama_model_loader: - type q5_K:  241 tensors
llama_model_loader: - type q6_K:   41 tensors
llm_load_vocab: special tokens definition check successful ( 259/32000 ).
llm_load_print_meta: format           = GGUF V2
llm_load_print_meta: arch             = llama
llm_load_print_meta: vocab type       = SPM
```

```
llm_load_print_meta: n_vocab          = 32000
llm_load_print_meta: n_merges         = 0
llm_load_print_meta: n_ctx_train      = 4096
llm_load_print_meta: n_embd           = 5120
llm_load_print_meta: n_head           = 40
llm_load_print_meta: n_head_kv        = 40
llm_load_print_meta: n_layer          = 40
llm_load_print_meta: n_rot            = 128
llm_load_print_meta: n_embd_head_k    = 128
llm_load_print_meta: n_embd_head_v    = 128
llm_load_print_meta: n_gqa            = 1
llm_load_print_meta: n_embd_k_gqa     = 5120
llm_load_print_meta: n_embd_v_gqa     = 5120
llm_load_print_meta: f_norm_eps       = 0.0e+00
llm_load_print_meta: f_norm_rms_eps   = 1.0e-05
llm_load_print_meta: f_clamp_kqv      = 0.0e+00
llm_load_print_meta: f_max_alibi_bias = 0.0e+00
llm_load_print_meta: f_logit_scale    = 0.0e+00
llm_load_print_meta: n_ff             = 13824
llm_load_print_meta: n_expert         = 0
llm_load_print_meta: n_expert_used    = 0
llm_load_print_meta: causal attn      = 1
llm_load_print_meta: pooling type     = 0
llm_load_print_meta: rope type        = 0
llm_load_print_meta: rope scaling     = linear
llm_load_print_meta: freq_base_train  = 10000.0
llm_load_print_meta: freq_scale_train = 1
llm_load_print_meta: n_yarn_orig_ctx  = 4096
llm_load_print_meta: rope_finetuned   = unknown
llm_load_print_meta: ssm_d_conv       = 0
llm_load_print_meta: ssm_d_inner      = 0
llm_load_print_meta: ssm_d_state      = 0
llm_load_print_meta: ssm_dt_rank      = 0
llm_load_print_meta: model type       = 13B
llm_load_print_meta: model ftype      = Q5_K - Medium
llm_load_print_meta: model params     = 13.02 B
llm_load_print_meta: model size       = 8.60 GiB (5.67 BPW)
llm_load_print_meta: general.name     = LLaMA v2
llm_load_print_meta: BOS token        = 1 '<s>'
llm_load_print_meta: EOS token        = 2 '</s>'
llm_load_print_meta: UNK token        = 0 '<unk>'
llm_load_print_meta: LF token         = 13 '<0x0A>'
llm_load_tensors: ggml ctx size =    0.37 MiB
llm_load_tensors: offloading 40 repeating layers to GPU
llm_load_tensors: offloading non-repeating layers to GPU
llm_load_tensors: offloaded 41/41 layers to GPU
llm_load_tensors:        CPU buffer size =   107.42 MiB
llm_load_tensors:      CUDA0 buffer size =  8694.21 MiB
...........................................................................................
..........
llama_new_context_with_model: n_ctx      = 4096
llama_new_context_with_model: n_batch    = 512
llama_new_context_with_model: n_ubatch   = 512
llama_new_context_with_model: flash_attn = 0
llama_new_context_with_model: freq_base  = 10000.0
llama_new_context_with_model: freq_scale = 1
llama_kv_cache_init:      CUDA0 KV buffer size =  3200.00 MiB
llama_new_context_with_model: KV self size  = 3200.00 MiB, K (f16): 1600.00 MiB, V (f16):
1600.00 MiB
llama_new_context_with_model:  CUDA_Host  output buffer size =     0.12 MiB
llama_new_context_with_model:      CUDA0 compute buffer size =   368.00 MiB
llama_new_context_with_model:  CUDA_Host compute buffer size =    18.01 MiB
llama_new_context_with_model: graph nodes  = 1286
llama_new_context_with_model: graph splits = 2
AVX = 1 | AVX_VNNI = 0 | AVX2 = 1 | AVX512 = 0 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | FMA
= 1 | NEON = 0 | ARM_FMA = 0 | F16C = 1 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 1 | SSE3
= 1 | SSSE3 = 1 | VSX = 0 | MATMUL_INT8 = 0 | LLAMAFILE = 1 |
Model metadata: {'tokenizer.ggml.unknown_token_id': '0', 'tokenizer.ggml.eos_token_id': '
2', 'general.architecture': 'llama', 'llama.context_length': '4096', 'general.name': 'LLa
MA v2', 'llama.embedding_length': '5120', 'llama.feed_forward_length': '13824', 'llama.at
tention.layer_norm_rms_epsilon': '0.000010', 'llama.rope.dimension_count': '128', 'llama.
attention.head_count': '40', 'tokenizer.ggml.bos_token_id': '1', 'llama.block_count': '40
', 'llama.attention.head_count_kv': '40', 'general.quantization_version': '2', 'tokenizer
.ggml.model': 'llama', 'general.file_type': '17'}
Using fallback chat format: None
```

Defining Model Response Parameters

In [ ]:

```python
def generate_llama_response(instruction, review):

    # System message explicitly instructing not to include the review text
    system_message = """
        [INST]<<SYS>>
        {}
        <</SYS>>[/INST]
    """.format(instruction)

    # Combine user_prompt and system_message to create the prompt
    prompt = f"{review}\n{system_message}"

    # Generate a response from the LLaMA model
    response = lcpp_llm(
        prompt=prompt,
        max_tokens=1024,
        temperature=0.01,
        top_p=0.95,
        repeat_penalty=1.2,
        top_k=50,
        stop=['INST'],
        echo=False,
        seed=42,
    )

    # Extract the sentiment from the response
    response_text = response["choices"][0]["text"]
    return response_text
```

- `max_tokens` : This parameter **specifies the maximum number of tokens that the model should generate** in response to the prompt.
- `temperature` : This parameter **controls the randomness of the generated response** . A higher temperature value will result in a more random response, while a lower temperature value will result in a more predictable response.
- `top_p` : This parameter **controls the diversity of the generated response by establishing a cumulative probability cutoff for token selection.** A higher value of top_p will result in a more diverse response, while a lower value will result in a less diverse response.
- `repeat_penalty` : This parameter **controls the penalty for repeating tokens in the generated response** . A higher value of repeat_penalty will result in a lower probability of repeating tokens, while a lower value will result in a higher probability of repeating tokens.
- `top_k` : This parameter **controls the maximum number of most-likely next tokens to consider** when generating the response at each step.
- `stop` : This parameter is a **list of tokens that are used to dynamically stop response generation** whenever the tokens in the list are encountered.
- `echo` : This parameter **controls whether the input (prompt) to the model should be returned** in the model response.
- `seed` : This parameter **specifies a seed value that helps replicate results** .

Task 1 - Ticket Categorization

Define the instruction for the task - Apply the generate response function to get an output from the model - Create a DataFrame containing the necessary fields from the model's output in a structured manner

In [ ]:

```
# create a copy of the data
data_1 = data.copy()
```

In [ ]:

```
## prompt to get the desired output
instruction_1 = """
[INST]<<SYS>>
You are acting as a guide for a technical assistant. As a technical assistant, your prima
ry task is to classify support tickets into specific categories. There are three categori
es:

Technical Issues
Hardware Issues
Data Recovery.
Please categorize the support ticket into the relevant category.

Here is an example:

support_ticket_text: My internet connection is very slow for the last 3 days. Therefore,
it is hard for me to work efficiently from home.
I am also facing frequent disconnections. Can you please help me to resolve this issue pr
omptly.
Please categorize the support ticket into one of the predefined categories (Technical Iss
ues, Hardware Issues, or Data Recovery). Other responses are not acceptable.
<</SYS>>[/INST]
"""
```

In [ ]:

```
# create a new column llama_response'
# by applying the generate_llama_response function to each ticket in the 'support_ticket_
text' column of the DataFrame 'data_1'
data_1['llama_response'] = data_1['support_ticket_text'].apply(lambda x: generate_llama_
response(instruction_1,x))
```

```
llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      30.14 ms /    53 runs   (    0.57 ms per tok
en,  1758.34 tokens per second)
llama_print_timings: prompt eval time =     967.80 ms /   260 tokens (    3.72 ms per tok
en,   268.65 tokens per second)
llama_print_timings:        eval time =    2761.65 ms /    52 runs   (   53.11 ms per tok
en,    18.83 tokens per second)
llama_print_timings:       total time =    3917.42 ms /   312 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      29.86 ms /    53 runs   (    0.56 ms per tok
en,  1774.89 tokens per second)
llama_print_timings: prompt eval time =     652.59 ms /   258 tokens (    2.53 ms per tok
en,   395.35 tokens per second)
llama_print_timings:        eval time =    2838.76 ms /    52 runs   (   54.59 ms per tok
en,    18.32 tokens per second)
llama_print_timings:       total time =    3669.71 ms /   310 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      30.66 ms /    52 runs   (    0.59 ms per tok
en,  1696.30 tokens per second)
llama_print_timings: prompt eval time =     544.57 ms /   254 tokens (    2.14 ms per tok
en,   466.42 tokens per second)
llama_print_timings:        eval time =    2861.89 ms /    51 runs   (   56.12 ms per tok
en,    17.82 tokens per second)
llama_print_timings:       total time =    3602.32 ms /   305 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      35.66 ms /    56 runs   (    0.64 ms per tok
en,  1570.30 tokens per second)
llama_print_timings: prompt eval time =     668.18 ms /   262 tokens (    2.55 ms per tok
en,   392.11 tokens per second)
llama_print_timings:        eval time =    2965.08 ms /    55 runs   (   53.91 ms per tok
en,    18.55 tokens per second)
llama_print_timings:       total time =    3884.73 ms /   317 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      30.81 ms /    54 runs   (    0.57 ms per tok
en,  1752.51 tokens per second)
llama_print_timings: prompt eval time =     530.85 ms /   237 tokens (    2.24 ms per tok
en,   446.45 tokens per second)
llama_print_timings:        eval time =    2979.31 ms /    53 runs   (   56.21 ms per tok
en,    17.79 tokens per second)
llama_print_timings:       total time =    3696.30 ms /   290 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      29.15 ms /    52 runs   (    0.56 ms per tok
en,  1784.12 tokens per second)
llama_print_timings: prompt eval time =     542.06 ms /   237 tokens (    2.29 ms per tok
en,   437.22 tokens per second)
llama_print_timings:        eval time =    2904.47 ms /    51 runs   (   56.95 ms per tok
en,    17.56 tokens per second)
llama_print_timings:       total time =    3611.78 ms /   288 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      32.37 ms /    52 runs   (    0.62 ms per tok
en,  1606.62 tokens per second)
llama_print_timings: prompt eval time =     550.23 ms /   240 tokens (    2.29 ms per tok
en,   436.18 tokens per second)
llama_print_timings:        eval time =    2867.35 ms /    51 runs   (   56.22 ms per tok
en,    17.79 tokens per second)
llama_print_timings:       total time =    3657.51 ms /   291 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      31.69 ms /    54 runs   (    0.59 ms per tok
en,  1703.79 tokens per second)
llama_print_timings: prompt eval time =     558.21 ms /   241 tokens (    2.32 ms per tok
en,   431.73 tokens per second)
llama_print_timings:        eval time =    3024.65 ms /    53 runs   (   57.07 ms per tok
en,    17.52 tokens per second)
llama_print_timings:       total time =    3780.53 ms /   294 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      29.70 ms /    52 runs   (    0.57 ms per tok
en,  1751.14 tokens per second)
llama_print_timings: prompt eval time =     546.08 ms /   235 tokens (    2.32 ms per tok
en,   430.34 tokens per second)
llama_print_timings:        eval time =    2910.99 ms /    51 runs   (   57.08 ms per tok
en,    17.52 tokens per second)
llama_print_timings:       total time =    3636.70 ms /   286 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      31.07 ms /    55 runs   (    0.56 ms per tok
en,  1769.91 tokens per second)
llama_print_timings: prompt eval time =     559.62 ms /   244 tokens (    2.29 ms per tok
en,   436.01 tokens per second)
llama_print_timings:        eval time =    3083.60 ms /    54 runs   (   57.10 ms per tok
en,    17.51 tokens per second)
llama_print_timings:       total time =    3834.54 ms /   298 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      33.88 ms /    53 runs   (    0.64 ms per tok
en,  1564.44 tokens per second)
llama_print_timings: prompt eval time =     553.89 ms /   237 tokens (    2.34 ms per tok
en,   427.88 tokens per second)
llama_print_timings:        eval time =    2888.53 ms /    52 runs   (   55.55 ms per tok
en,    18.00 tokens per second)
llama_print_timings:       total time =    3708.48 ms /   289 tokens
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      29.32 ms /    52 runs   (    0.56 ms per tok
en,  1773.35 tokens per second)
llama_print_timings: prompt eval time =     565.40 ms /   241 tokens (    2.35 ms per tok
en,   426.24 tokens per second)
llama_print_timings:        eval time =    2921.53 ms /    51 runs   (   57.28 ms per tok
en,    17.46 tokens per second)
llama_print_timings:       total time =    3668.70 ms /   292 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      29.57 ms /    52 runs   (    0.57 ms per tok
en,  1758.30 tokens per second)
llama_print_timings: prompt eval time =     567.97 ms /   241 tokens (    2.36 ms per tok
en,   424.32 tokens per second)
llama_print_timings:        eval time =    2928.57 ms /    51 runs   (   57.42 ms per tok
en,    17.41 tokens per second)
llama_print_timings:       total time =    3674.93 ms /   292 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      32.39 ms /    52 runs   (    0.62 ms per tok
en,  1605.43 tokens per second)
llama_print_timings: prompt eval time =     557.62 ms /   233 tokens (    2.39 ms per tok
en,   417.85 tokens per second)
llama_print_timings:        eval time =    2862.34 ms /    51 runs   (   56.12 ms per tok
en,    17.82 tokens per second)
llama_print_timings:       total time =    3653.11 ms /   284 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      34.12 ms /    59 runs   (    0.58 ms per tok
en,  1729.19 tokens per second)
llama_print_timings: prompt eval time =     572.91 ms /   238 tokens (    2.41 ms per tok
en,   415.42 tokens per second)
llama_print_timings:        eval time =    3293.91 ms /    58 runs   (   56.79 ms per tok
en,    17.61 tokens per second)
llama_print_timings:       total time =    4097.35 ms /   296 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      31.56 ms /    56 runs   (    0.56 ms per tok
en,  1774.40 tokens per second)
llama_print_timings: prompt eval time =     552.99 ms /   229 tokens (    2.41 ms per tok
en,   414.11 tokens per second)
llama_print_timings:        eval time =    3147.41 ms /    55 runs   (   57.23 ms per tok
en,    17.47 tokens per second)
llama_print_timings:       total time =    3892.71 ms /   284 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      30.69 ms /    54 runs   (    0.57 ms per tok
en,  1759.42 tokens per second)
llama_print_timings: prompt eval time =     558.03 ms /   234 tokens (    2.38 ms per tok
en,   419.33 tokens per second)
llama_print_timings:        eval time =    3070.41 ms /    53 runs   (   57.93 ms per tok
en,    17.26 tokens per second)
llama_print_timings:       total time =    3813.62 ms /   287 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      32.66 ms /    52 runs   (    0.63 ms per tok
en,  1592.11 tokens per second)
llama_print_timings: prompt eval time =     547.47 ms /   229 tokens (    2.39 ms per tok
en,   418.29 tokens per second)
llama_print_timings:        eval time =    2846.27 ms /    51 runs   (   55.81 ms per tok
en,    17.92 tokens per second)
llama_print_timings:       total time =    3652.05 ms /   280 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      28.65 ms /    52 runs   (    0.55 ms per tok
en,  1814.76 tokens per second)
llama_print_timings: prompt eval time =     564.58 ms /   230 tokens (    2.45 ms per tok
en,   407.38 tokens per second)
llama_print_timings:        eval time =    2954.70 ms /    51 runs   (   57.94 ms per tok
en,    17.26 tokens per second)
llama_print_timings:       total time =    3690.74 ms /   281 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      30.37 ms /    53 runs   (    0.57 ms per tok
en,  1745.26 tokens per second)
llama_print_timings: prompt eval time =     839.55 ms /   386 tokens (    2.17 ms per tok
en,   459.77 tokens per second)
llama_print_timings:        eval time =    3041.82 ms /    52 runs   (   58.50 ms per tok
en,    17.10 tokens per second)
llama_print_timings:       total time =    4076.66 ms /   438 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      43.64 ms /    59 runs   (    0.74 ms per tok
en,  1352.00 tokens per second)
llama_print_timings: prompt eval time =     969.38 ms /   499 tokens (    1.94 ms per tok
en,   514.76 tokens per second)
llama_print_timings:        eval time =    3221.01 ms /    58 runs   (   55.53 ms per tok
en,    18.01 tokens per second)
llama_print_timings:       total time =    4505.08 ms /   557 tokens
```

In [ ]:

```
data_1['llama_response'][0]
```

Out[ ]:

```
' Based on the information provided in the support ticket, I would categorize it as a Tec
hnical Issue. The slow internet connection and frequent disconnections suggest a problem
with the connectivity or network configuration, which falls under the umbrella of technic
al issues.'
```

In [ ]:

```
# check the first five rows of the data to confirm whether the new column has been added
data_1.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	llama_response
0	ST2023-006	My internet connection has significantly slowe...	Based on the information provided in the supp...
1	ST2023-007	Urgent help required! My laptop refuses to sta...	Based on the information provided in the supp...
2	ST2023-008	I've accidentally deleted essential work docum...	Based on the information provided in the supp...
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	Based on the information provided in the supp...
4	ST2023-010	My smartphone battery is draining rapidly, eve...	Sure! Based on the information provided in th...

In [ ]:

```
print(len(data_1['llama_response']))
```

```
21
```

In [ ]:

```
def extract_category(model_response):
    if 'technical issues' in model_response.lower():
        return 'Technical issues'
    elif 'hardware issues' in model_response.lower():
        return 'Hardware issues'
    elif 'data recovery' in model_response.lower():
        return 'Data recovery'
```

In [ ]:

```
data_1['Category'] = data_1['llama_response'].apply(extract_category)
data_1['Category'].head()
```

Out[ ]:

```
0    Technical issues
1    Technical issues
2    Technical issues
3       Data recovery
4    Technical issues
Name: Category, dtype: object
```

In [ ]:

```
final_data_1 = data_1.drop(['llama_response'], axis=1)
final_data_1.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	Category
0	ST2023-006	My internet connection has significantly slowe...	Technical issues
1	ST2023-007	Urgent help required! My laptop refuses to sta...	Technical issues
2	ST2023-008	I've accidentally deleted essential work docum...	Technical issues
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	Data recovery
4	ST2023-010	My smartphone battery is draining rapidly, eve...	Technical issues

Task 2 - Ticket Categorization and Returning Structured Output

Define the instruction for the task - Apply the generate response function to get an output from the model - Create a DataFrame containing the necessary fields from the model's output in a structured manner

In [ ]:

```
# create a copy of the data
data_2 = data.copy()
```

In [ ]:

```
## prompt to get the desired output
instruction_2 = """
[INST]<<SYS>>
You are acting as a guide for a technical assistant. As a technical assistant, your prima
ry task is to classify support tickets into specific categories. There are three categori
es:

Technical Issues
Hardware Issues
Data Recovery.

{"category": "<create one of the predefined categories (Technical Issues, Hardware Issues
, and Data Recovery). Other responses are not acceptable>"}

Please categorize the support ticket into the relevant category.

Here is an example:

support_ticket_text: My internet connection is very slow for the last 3 days. Therefore,
it is hard for me to work efficiently from home.
I am also facing frequent disconnections. Can you please help me to resolve this issue pr
omptly.
```

```
Format the output as a JSON object with a single key-value pair as shown below:
{"category": "Technical Issues"}

Do not include any other text in the output except the JSON.
<</SYS>>[/INST]
"""
```

In [ ]:

```
# create a new column llama_response'
# by applying the generate_llama_response function to each ticket in the 'support_ticket_
text' column of the DataFrame 'data_2'
data_2['llama_response'] = data_2['support_ticket_text'].apply(lambda x: generate_llama_
response(instruction_2,x))
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       9.38 ms /    17 runs   (    0.55 ms per tok
en,  1812.37 tokens per second)
llama_print_timings: prompt eval time =     732.42 ms /   306 tokens (    2.39 ms per tok
en,   417.79 tokens per second)
llama_print_timings:        eval time =     925.01 ms /    16 runs   (   57.81 ms per tok
en,    17.30 tokens per second)
llama_print_timings:       total time =    1713.84 ms /   322 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       8.97 ms /    17 runs   (    0.53 ms per tok
en,  1894.15 tokens per second)
llama_print_timings: prompt eval time =     755.15 ms /   305 tokens (    2.48 ms per tok
en,   403.89 tokens per second)
llama_print_timings:        eval time =     912.30 ms /    16 runs   (   57.02 ms per tok
en,    17.54 tokens per second)
llama_print_timings:       total time =    1725.42 ms /   321 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       4.88 ms /     9 runs   (    0.54 ms per to
ken,  1845.40 tokens per second)
llama_print_timings: prompt eval time =     756.11 ms /   301 tokens (    2.51 ms per tok
en,   398.09 tokens per second)
llama_print_timings:        eval time =     452.97 ms /     8 runs   (   56.62 ms per to
ken,    17.66 tokens per second)
llama_print_timings:       total time =    1240.61 ms /   309 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       6.02 ms /    11 runs   (    0.55 ms per tok
en,  1828.76 tokens per second)
llama_print_timings: prompt eval time =     771.99 ms /   309 tokens (    2.50 ms per tok
en,   400.26 tokens per second)
llama_print_timings:        eval time =     573.43 ms /    10 runs   (   57.34 ms per tok
en,    17.44 tokens per second)
llama_print_timings:       total time =    1382.36 ms /   319 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       6.95 ms /    11 runs   (    0.63 ms per tok
en,  1583.19 tokens per second)
llama_print_timings: prompt eval time =     739.88 ms /   284 tokens (    2.61 ms per tok
en,   383.84 tokens per second)
llama_print_timings:        eval time =     560.46 ms /    10 runs   (   56.05 ms per tok
en,    17.84 tokens per second)
llama_print_timings:       total time =    1357.00 ms /   294 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       5.68 ms /     9 runs   (    0.63 ms per to
ken,  1585.62 tokens per second)
llama_print_timings: prompt eval time =     756.45 ms /   284 tokens (    2.66 ms per tok
en,   375.44 tokens per second)
```

```
llama_print_timings:        eval time =     440.19 ms /     8 runs   (   55.02 ms per to
ken,    18.17 tokens per second)
llama_print_timings:       total time =    1242.91 ms /   292 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       5.77 ms /    11 runs   (    0.52 ms per tok
en,  1907.73 tokens per second)
llama_print_timings: prompt eval time =     750.62 ms /   287 tokens (    2.62 ms per tok
en,   382.35 tokens per second)
llama_print_timings:        eval time =     584.57 ms /    10 runs   (   58.46 ms per tok
en,    17.11 tokens per second)
llama_print_timings:       total time =    1372.39 ms /   297 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       5.81 ms /    11 runs   (    0.53 ms per tok
en,  1894.59 tokens per second)
llama_print_timings: prompt eval time =     767.18 ms /   288 tokens (    2.66 ms per tok
en,   375.40 tokens per second)
llama_print_timings:        eval time =     590.93 ms /    10 runs   (   59.09 ms per tok
en,    16.92 tokens per second)
llama_print_timings:       total time =    1393.06 ms /   298 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       5.77 ms /    11 runs   (    0.52 ms per tok
en,  1905.42 tokens per second)
llama_print_timings: prompt eval time =     755.63 ms /   282 tokens (    2.68 ms per tok
en,   373.20 tokens per second)
llama_print_timings:        eval time =     604.37 ms /    10 runs   (   60.44 ms per tok
en,    16.55 tokens per second)
llama_print_timings:       total time =    1395.67 ms /   292 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       7.01 ms /    13 runs   (    0.54 ms per tok
en,  1855.02 tokens per second)
llama_print_timings: prompt eval time =     773.31 ms /   291 tokens (    2.66 ms per tok
en,   376.30 tokens per second)
llama_print_timings:        eval time =     717.65 ms /    12 runs   (   59.80 ms per tok
en,    16.72 tokens per second)
llama_print_timings:       total time =    1532.81 ms /   303 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       5.03 ms /     9 runs   (    0.56 ms per to
ken,  1788.20 tokens per second)
llama_print_timings: prompt eval time =     759.96 ms /   284 tokens (    2.68 ms per tok
en,   373.70 tokens per second)
llama_print_timings:        eval time =     481.48 ms /     8 runs   (   60.19 ms per to
ken,    16.62 tokens per second)
llama_print_timings:       total time =    1273.77 ms /   292 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       5.93 ms /    11 runs   (    0.54 ms per tok
en,  1855.91 tokens per second)
llama_print_timings: prompt eval time =     769.52 ms /   288 tokens (    2.67 ms per tok
en,   374.26 tokens per second)
llama_print_timings:        eval time =     591.70 ms /    10 runs   (   59.17 ms per tok
en,    16.90 tokens per second)
llama_print_timings:       total time =    1398.65 ms /   298 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       5.83 ms /    11 runs   (    0.53 ms per tok
en,  1886.47 tokens per second)
llama_print_timings: prompt eval time =     768.76 ms /   288 tokens (    2.67 ms per tok
en,   374.63 tokens per second)
llama_print_timings:        eval time =     608.57 ms /    10 runs   (   60.86 ms per tok
en,    16.43 tokens per second)
```

```
llama_print_timings:       total time =    1412.16 ms /   298 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       5.97 ms /     9 runs   (    0.66 ms per to
ken,  1506.78 tokens per second)
llama_print_timings: prompt eval time =     782.77 ms /   280 tokens (    2.80 ms per tok
en,   357.70 tokens per second)
llama_print_timings:        eval time =     465.30 ms /     8 runs   (   58.16 ms per to
ken,    17.19 tokens per second)
llama_print_timings:       total time =    1293.13 ms /   288 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       7.15 ms /    11 runs   (    0.65 ms per tok
en,  1538.46 tokens per second)
llama_print_timings: prompt eval time =     775.09 ms /   285 tokens (    2.72 ms per tok
en,   367.70 tokens per second)
llama_print_timings:        eval time =     570.92 ms /    10 runs   (   57.09 ms per tok
en,    17.52 tokens per second)
llama_print_timings:       total time =    1405.07 ms /   295 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       6.98 ms /    11 runs   (    0.63 ms per tok
en,  1576.38 tokens per second)
llama_print_timings: prompt eval time =     774.96 ms /   276 tokens (    2.81 ms per tok
en,   356.15 tokens per second)
llama_print_timings:        eval time =     571.20 ms /    10 runs   (   57.12 ms per tok
en,    17.51 tokens per second)
llama_print_timings:       total time =    1400.74 ms /   286 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       5.81 ms /    11 runs   (    0.53 ms per tok
en,  1894.59 tokens per second)
llama_print_timings: prompt eval time =     780.61 ms /   281 tokens (    2.78 ms per tok
en,   359.98 tokens per second)
llama_print_timings:        eval time =     599.03 ms /    10 runs   (   59.90 ms per tok
en,    16.69 tokens per second)
llama_print_timings:       total time =    1415.33 ms /   291 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       5.67 ms /     9 runs   (    0.63 ms per to
ken,  1587.30 tokens per second)
llama_print_timings: prompt eval time =     780.28 ms /   276 tokens (    2.83 ms per tok
en,   353.72 tokens per second)
llama_print_timings:        eval time =     484.57 ms /     8 runs   (   60.57 ms per to
ken,    16.51 tokens per second)
llama_print_timings:       total time =    1300.54 ms /   284 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       4.74 ms /     9 runs   (    0.53 ms per to
ken,  1897.53 tokens per second)
llama_print_timings: prompt eval time =     786.06 ms /   277 tokens (    2.84 ms per tok
en,   352.39 tokens per second)
llama_print_timings:        eval time =     478.38 ms /     8 runs   (   59.80 ms per to
ken,    16.72 tokens per second)
llama_print_timings:       total time =    1296.38 ms /   285 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       8.39 ms /    15 runs   (    0.56 ms per tok
en,  1786.78 tokens per second)
llama_print_timings: prompt eval time =     922.76 ms /   433 tokens (    2.13 ms per tok
en,   469.25 tokens per second)
llama_print_timings:        eval time =     829.78 ms /    14 runs   (   59.27 ms per tok
en,    16.87 tokens per second)
llama_print_timings:       total time =    1808.69 ms /   447 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =       7.27 ms /    13 runs   (    0.56 ms per tok
en,  1788.17 tokens per second)
llama_print_timings: prompt eval time =    1394.69 ms /   546 tokens (    2.55 ms per tok
en,   391.49 tokens per second)
llama_print_timings:        eval time =     752.66 ms /    12 runs   (   62.72 ms per tok
en,    15.94 tokens per second)
llama_print_timings:       total time =    2196.44 ms /   558 tokens
```

In [ ]:

```
# check the first five rows of the data to confirm whether the new column has been added
data_2['llama_response'].head()
```

Out[ ]:

```
0     {\n       "category": "Technical Issues"\n   ...
1     {\n       "category": "Hardware Issues"\n      }
2                        {"category": "Data Recovery"}
3                      {"category": "Hardware Issues"}
4                      {"category": "Hardware Issues"}
Name: llama_response, dtype: object
```

In [ ]:

```
# defining a function to parse the JSON output from the model
def extract_json_data(json_str):
    try:
        # Find the indices of the opening and closing curly braces
        json_start = json_str.find('{')
        json_end = json_str.rfind('}')

        if json_start != -1 and json_end != -1:
            extracted_category = json_str[json_start:json_end + 1]  # Extract the JSON o
bject
            data_dict = json.loads(extracted_category)
            return data_dict
        else:
            print(f"Warning: JSON object not found in response: {json_str}")
            return {}
    except json.JSONDecodeError as e:
        print(f"Error parsing JSON: {e}")
        return {}
```

In [ ]:

```
# apply the extract_json_data function on the llama_response column to create a new colum
n called llama_response_parsed
data_2['llama_response_parsed'] = data_2['llama_response'].apply(extract_json_data)
data_2['llama_response_parsed'].head()
```

Out[ ]:

```
0    {'category': 'Technical Issues'}
1     {'category': 'Hardware Issues'}
2       {'category': 'Data Recovery'}
3     {'category': 'Hardware Issues'}
4     {'category': 'Hardware Issues'}
Name: llama_response_parsed, dtype: object
```

In [ ]:

```
# apply the json_normalize on llama_response_parsed variable
llama_response_parsed_df_2 = pd.json_normalize(data_2['llama_response_parsed'])
llama_response_parsed_df_2.head()
```

Out[ ]:

	category
0	Technical Issues

	category
1	Hardware Issues
2	Data Recovery
3	Hardware Issues
4	Hardware Issues

In [ ]:

```
# concat data_2 and llama_response_parsed_df_2
data_with_parsed_model_output_2 = pd.concat([data_2, llama_response_parsed_df_2], axis=1
)
data_with_parsed_model_output_2.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	llama_response	llama_response_parsed	category
0	ST2023-006	My internet connection has significantly slowe...	{\n "category": "Technical Issues"\n ...	{'category': 'Technical Issues'}	Technical Issues
1	ST2023-007	Urgent help required! My laptop refuses to sta...	{\n "category": "Hardware Issues"\n }	{'category': 'Hardware Issues'}	Hardware Issues
2	ST2023-008	I've accidentally deleted essential work docum...	{"category": "Data Recovery"}	{'category': 'Data Recovery'}	Data Recovery
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	{"category": "Hardware Issues"}	{'category': 'Hardware Issues'}	Hardware Issues
4	ST2023-010	My smartphone battery is draining rapidly, eve...	{"category": "Hardware Issues"}	{'category': 'Hardware Issues'}	Hardware Issues

In [ ]:

```
## drop llama_response and llama_response_parsed variables
final_data_2 = data_with_parsed_model_output_2.drop(['llama_response','llama_response_par
sed'], axis=1)
final_data_2.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	category
0	ST2023-006	My internet connection has significantly slowe...	Technical Issues
1	ST2023-007	Urgent help required! My laptop refuses to sta...	Hardware Issues
2	ST2023-008	I've accidentally deleted essential work docum...	Data Recovery
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	Hardware Issues
4	ST2023-010	My smartphone battery is draining rapidly, eve...	Hardware Issues

Task 3 - Ticket Categorization, Creating Tags, and Returning Structured Output

Define the instruction for the task - Apply the generate response function to get an output from the model - Create a DataFrame containing the necessary fields from the model's output in a structured manner

In [ ]:

```
# create a copy of the data
data_3 = data.copy()
```

In [ ]:

```
## prompt to get the desired output
instruction_3 = """

[INST]<<SYS>>
You are acting as a guide for a technical assistant. As a technical assistant, your prima
```

```
ry task is to classify the support ticket into specific categories. There are three categ
ories:

Technical Issues
Hardware Issues
Data Recovery

As a technical assistant, you should only respond with one of the predefined categories (
Technical Issues, Hardware Issues, and Data Recovery). Other responses are not acceptable
.

Your goal is to identify the category and then provide the following information from the
support ticket text, including the creating tags. Go through each support ticket text tho
roughly and considering the overall sentiment before responding, and generate only a stru
ctured output for the JSON format.

Here is the structured output for further analysis in JSON format. Follow the JSON format
below strictly. You must include the following as JSON output:

{"category": <create one of the predefined categories (Technical Issues, Hardware Issues,
and Data Recovery). Other responses are not acceptable>,
"tags": <create tags to further classify the ticket>
}

Here is an example:

support_ticket_text: My internet connection has significantly slowed down over the past t
wo days, making it challenging to work efficiently from home. Frequent disconnections are
causing major disruptions. Please assist in resolving this connectivity issue promptly.

Your output should be:

{"category": "Technical Issues",
"tags": ["internet connectivity", "slowdown", "frequent disconnections"]}


Do not include any other text in the output except the JSON.

<</SYS>>[/INST]
    """
```

In [ ]:

```
# create a new column llama_response'
# by applying the generate_llama_response function to each ticket in the 'support_ticket_
text' column of the DataFrame 'data_3'
data_3['llama_response'] = data_3['support_ticket_text'].apply(lambda x: generate_llama_
response(instruction_3,x))
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      18.04 ms /    34 runs   (    0.53 ms per tok
en,  1885.01 tokens per second)
llama_print_timings: prompt eval time =     867.20 ms /   447 tokens (    1.94 ms per tok
en,   515.45 tokens per second)
llama_print_timings:        eval time =    1771.78 ms /    33 runs   (   53.69 ms per tok
en,    18.63 tokens per second)
llama_print_timings:       total time =    2756.92 ms /   480 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      15.78 ms /    30 runs   (    0.53 ms per tok
en,  1900.78 tokens per second)
llama_print_timings: prompt eval time =     813.77 ms /   446 tokens (    1.82 ms per tok
en,   548.07 tokens per second)
llama_print_timings:        eval time =    1564.75 ms /    29 runs   (   53.96 ms per tok
en,    18.53 tokens per second)
llama_print_timings:       total time =    2474.72 ms /   475 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
```

```
llama_print_timings:      sample time =      50.62 ms /    87 runs   (    0.58 ms per tok
en,  1718.79 tokens per second)
llama_print_timings: prompt eval time =     820.10 ms /   442 tokens (    1.86 ms per tok
en,   538.96 tokens per second)
llama_print_timings:        eval time =    4787.59 ms /    86 runs   (   55.67 ms per tok
en,    17.96 tokens per second)
llama_print_timings:       total time =    5937.38 ms /   528 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      19.53 ms /    31 runs   (    0.63 ms per tok
en,  1587.55 tokens per second)
llama_print_timings: prompt eval time =     828.40 ms /   450 tokens (    1.84 ms per tok
en,   543.21 tokens per second)
llama_print_timings:        eval time =    1605.14 ms /    30 runs   (   53.50 ms per tok
en,    18.69 tokens per second)
llama_print_timings:       total time =    2592.04 ms /   480 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      46.33 ms /    82 runs   (    0.57 ms per tok
en,  1769.87 tokens per second)
llama_print_timings: prompt eval time =     810.77 ms /   425 tokens (    1.91 ms per tok
en,   524.19 tokens per second)
llama_print_timings:        eval time =    4617.62 ms /    81 runs   (   57.01 ms per tok
en,    17.54 tokens per second)
llama_print_timings:       total time =    5722.13 ms /   506 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      16.78 ms /    29 runs   (    0.58 ms per tok
en,  1727.84 tokens per second)
llama_print_timings: prompt eval time =     802.45 ms /   425 tokens (    1.89 ms per tok
en,   529.63 tokens per second)
llama_print_timings:        eval time =    1595.48 ms /    28 runs   (   56.98 ms per tok
en,    17.55 tokens per second)
llama_print_timings:       total time =    2506.42 ms /   453 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      18.23 ms /    31 runs   (    0.59 ms per tok
en,  1700.59 tokens per second)
llama_print_timings: prompt eval time =     827.77 ms /   428 tokens (    1.93 ms per tok
en,   517.05 tokens per second)
llama_print_timings:        eval time =    1678.89 ms /    30 runs   (   55.96 ms per tok
en,    17.87 tokens per second)
llama_print_timings:       total time =    2658.22 ms /   458 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      17.79 ms /    31 runs   (    0.57 ms per tok
en,  1742.55 tokens per second)
llama_print_timings: prompt eval time =     842.16 ms /   429 tokens (    1.96 ms per tok
en,   509.41 tokens per second)
llama_print_timings:        eval time =    1690.21 ms /    30 runs   (   56.34 ms per tok
en,    17.75 tokens per second)
llama_print_timings:       total time =    2669.73 ms /   459 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      17.03 ms /    32 runs   (    0.53 ms per tok
en,  1879.04 tokens per second)
llama_print_timings: prompt eval time =     829.57 ms /   423 tokens (    1.96 ms per tok
en,   509.90 tokens per second)
llama_print_timings:        eval time =    1777.50 ms /    31 runs   (   57.34 ms per tok
en,    17.44 tokens per second)
llama_print_timings:       total time =    2722.91 ms /   454 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      18.78 ms /    36 runs   (    0.52 ms per tok
en,  1916.93 tokens per second)
```

```
llama_print_timings: prompt eval time =     839.70 ms /   432 tokens (    1.94 ms per tok
en,   514.47 tokens per second)
llama_print_timings:        eval time =    2039.86 ms /    35 runs   (   58.28 ms per tok
en,    17.16 tokens per second)
llama_print_timings:       total time =    2998.80 ms /   467 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      49.98 ms /    82 runs   (    0.61 ms per tok
en,  1640.56 tokens per second)
llama_print_timings: prompt eval time =     844.41 ms /   425 tokens (    1.99 ms per tok
en,   503.31 tokens per second)
llama_print_timings:        eval time =    4567.29 ms /    81 runs   (   56.39 ms per tok
en,    17.73 tokens per second)
llama_print_timings:       total time =    5777.05 ms /   506 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      13.62 ms /    26 runs   (    0.52 ms per tok
en,  1908.26 tokens per second)
llama_print_timings: prompt eval time =     843.93 ms /   429 tokens (    1.97 ms per tok
en,   508.34 tokens per second)
llama_print_timings:        eval time =    1451.75 ms /    25 runs   (   58.07 ms per tok
en,    17.22 tokens per second)
llama_print_timings:       total time =    2384.46 ms /   454 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      43.18 ms /    79 runs   (    0.55 ms per tok
en,  1829.51 tokens per second)
llama_print_timings: prompt eval time =     854.81 ms /   429 tokens (    1.99 ms per tok
en,   501.87 tokens per second)
llama_print_timings:        eval time =    4471.14 ms /    78 runs   (   57.32 ms per tok
en,    17.45 tokens per second)
llama_print_timings:       total time =    5600.46 ms /   507 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      18.88 ms /    35 runs   (    0.54 ms per tok
en,  1853.91 tokens per second)
llama_print_timings: prompt eval time =     851.35 ms /   421 tokens (    2.02 ms per tok
en,   494.51 tokens per second)
llama_print_timings:        eval time =    1985.03 ms /    34 runs   (   58.38 ms per tok
en,    17.13 tokens per second)
llama_print_timings:       total time =    2962.53 ms /   455 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      44.57 ms /    73 runs   (    0.61 ms per tok
en,  1637.84 tokens per second)
llama_print_timings: prompt eval time =     862.01 ms /   426 tokens (    2.02 ms per tok
en,   494.20 tokens per second)
llama_print_timings:        eval time =    4076.82 ms /    72 runs   (   56.62 ms per tok
en,    17.66 tokens per second)
llama_print_timings:       total time =    5274.49 ms /   498 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      19.59 ms /    37 runs   (    0.53 ms per tok
en,  1888.43 tokens per second)
llama_print_timings: prompt eval time =     858.90 ms /   417 tokens (    2.06 ms per tok
en,   485.50 tokens per second)
llama_print_timings:        eval time =    2108.27 ms /    36 runs   (   58.56 ms per tok
en,    17.08 tokens per second)
llama_print_timings:       total time =    3095.60 ms /   453 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      19.44 ms /    37 runs   (    0.53 ms per tok
en,  1902.90 tokens per second)
llama_print_timings: prompt eval time =     870.56 ms /   422 tokens (    2.06 ms per tok
en,   484.74 tokens per second)
```

```
llama_print_timings:        eval time =    2136.24 ms /    36 runs   (   59.34 ms per tok
en,    16.85 tokens per second)
llama_print_timings:       total time =    3131.68 ms /   458 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      22.13 ms /    33 runs   (    0.67 ms per tok
en,  1491.32 tokens per second)
llama_print_timings: prompt eval time =     872.10 ms /   417 tokens (    2.09 ms per tok
en,   478.16 tokens per second)
llama_print_timings:        eval time =    1835.26 ms /    32 runs   (   57.35 ms per tok
en,    17.44 tokens per second)
llama_print_timings:       total time =    2864.35 ms /   449 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      19.09 ms /    33 runs   (    0.58 ms per tok
en,  1728.65 tokens per second)
llama_print_timings: prompt eval time =     873.80 ms /   418 tokens (    2.09 ms per tok
en,   478.37 tokens per second)
llama_print_timings:        eval time =    1859.88 ms /    32 runs   (   58.12 ms per tok
en,    17.21 tokens per second)
llama_print_timings:       total time =    2877.43 ms /   450 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      21.53 ms /    34 runs   (    0.63 ms per tok
en,  1578.97 tokens per second)
llama_print_timings: prompt eval time =    1402.16 ms /   574 tokens (    2.44 ms per tok
en,   409.37 tokens per second)
llama_print_timings:        eval time =    1937.36 ms /    33 runs   (   58.71 ms per tok
en,    17.03 tokens per second)
llama_print_timings:       total time =    3521.47 ms /   607 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      17.41 ms /    32 runs   (    0.54 ms per tok
en,  1837.60 tokens per second)
llama_print_timings: prompt eval time =    1609.00 ms /   687 tokens (    2.34 ms per tok
en,   426.97 tokens per second)
llama_print_timings:        eval time =    1919.89 ms /    31 runs   (   61.93 ms per tok
en,    16.15 tokens per second)
llama_print_timings:       total time =    3656.27 ms /   718 tokens
```

In [ ]:

```
data_3['llama_response'][0]
```

Out[ ]:

```
'\n{"category": "Technical Issues",\n"tags": ["internet connectivity", "slowdown", "frequ
ent disconnections"]}'
```

In [ ]:

```
# check the first five rows of the data to confirm whether the new column has been added
data_3['llama_response'].head()
```

Out[ ]:

```
0    \n{"category": "Technical Issues",\n"tags": ["...
1    \n{"category": "Hardware Issues",\n"tags": ["l...
2    \n    Based on the information provided, I wou...
3    \n{"category": "Technical Issues",\n"tags": ["...
4    \n    Sure, I'd be happy to help! Based on you...
Name: llama_response, dtype: object
```

In [ ]:

```
# apply the extract_json_data function on the llama_response column to create a new colum
n called llama_response_parsed
data_3['llama_response_parsed'] = data_3['llama_response'].apply(extract_json_data)
```

```
data_3['llama_response_parsed'].head()
```

Out[ ]:

```
0    {'category': 'Technical Issues', 'tags': ['int...
1    {'category': 'Hardware Issues', 'tags': ['lapt...
2    {'category': 'Data Recovery', 'tags': ['data l...
3    {'category': 'Technical Issues', 'tags': ['wea...
4    {'category': 'Technical Issues', 'tags': ['sma...
Name: llama_response_parsed, dtype: object
```

In [ ]:

```
# apply the json_normalize on llama_response_parsed variable
llama_response_parsed_df_3 = pd.json_normalize(data_3['llama_response_parsed'])
llama_response_parsed_df_3.head()
```

Out[ ]:

	category	tags
0	Technical Issues	[internet connectivity, slowdown, frequent dis...
1	Hardware Issues	[laptop, start-up, restart]
2	Data Recovery	[data loss, documents, accidental deletion]
3	Technical Issues	[weak Wi-Fi signal, proximity to router]
4	Technical Issues	[smartphone battery drain, rapid draining, min...

In [ ]:

```
# concat data_3 and llama_response_parsed_df_3
data_with_parsed_model_output_3 = pd.concat([data_3, llama_response_parsed_df_3], axis=1
)
data_with_parsed_model_output_3.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	llama_response	llama_response_parsed	category	tags
0	ST2023-006	My internet connection has significantly slowe...	\n{"category": "Technical Issues",\n"tags": ["...	{'category': 'Technical Issues', 'tags': ['int...	Technical Issues	[internet connectivity, slowdown, frequent dis...
1	ST2023-007	Urgent help required! My laptop refuses to sta...	\n{"category": "Hardware Issues",\n"tags": ["l...	{'category': 'Hardware Issues', 'tags': ['lapt...	Hardware Issues	[laptop, start-up, restart]
2	ST2023-008	I've accidentally deleted essential work docum...	\n Based on the information provided, I wou...	{'category': 'Data Recovery', 'tags': ['data l...	Data Recovery	[data loss, documents, accidental deletion]
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	\n{"category": "Technical Issues",\n"tags": ["...	{'category': 'Technical Issues', 'tags': ['wea...	Technical Issues	[weak Wi-Fi signal, proximity to router]
4	ST2023-010	My smartphone battery is draining rapidly, eve...	\n Sure, I'd be happy to help! Based on you...	{'category': 'Technical Issues', 'tags': ['sma...	Technical Issues	[smartphone battery drain, rapid draining, min...

In [ ]:

```
# drop llama_response and llama_response_parsed variables
final_data_3 = data_with_parsed_model_output_3.drop(["llama_response","llama_response_par
sed"], axis=1)
final_data_3.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	category	tags
0	ST2023-006	My internet connection has significantly	Technical	[internet connectivity, slowdown, frequent

	support_tick_id	support_ticket_text	category	tags
0	ST2023-006	slowe	Issues	dis
1	ST2023-007	Urgent help required! My laptop refuses to sta...	Hardware Issues	[laptop, start-up, restart]
2	ST2023-008	I've accidentally deleted essential work docum...	Data Recovery	[data loss, documents, accidental deletion]
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	Technical Issues	[weak Wi-Fi signal, proximity to router]
4	ST2023-010	My smartphone battery is draining rapidly, eve...	Technical Issues	[smartphone battery drain, rapid draining, min...

Task 4 - Ticket Categorization, Creating Tags, Assigning Priority, and Returning Structured Output

Define the instruction for the task - Apply the generate response function to get an output from the model - Create a DataFrame containing the necessary fields from the model's output in a structured manner

In [ ]:

```
# create a copy of the data
data_4 = data.copy()
```

In [ ]:

```
## prompt to get the desired output
instruction_4 = """

    [INST]<<SYS>>
You are acting as a guide for a technical assistant. As a technical assistant, your prima
ry task is to classify the support ticket into specific categories. There are three categ
ories:

Technical Issues
Hardware Issues
Data Recovery

As a technical assistant, you should only respond with one of the predefined categories (
Technical Issues, Hardware Issues, and Data Recovery). Other responses are not acceptable
.

Your goal is to identify the category and then provide the following information from the
support ticket text, including the creating tags, assigning priority. Go through each sup
port ticket text thoroughly and considering the overall sentiment before responding, and
generate only a structured output for the JSON format.

Here is the structured output for further analysis in JSON format. Follow the JSON format
below strictly. You must include the following as JSON output:

{"category": <create one of the predefined categories (Technical Issues, Hardware Issues,
and Data Recovery). Other responses are not acceptable>,
"tags": <create tags to further classify the ticket>,
"priority": <assign a priority level (e.g., "High" or "Normal") only based on the underst
anding of the text>
}

Here is an example:

support_ticket_text: My internet connection has significantly slowed down over the past t
wo days, making it challenging to work efficiently from home. Frequent disconnections are
causing major disruptions. Please assist in resolving this connectivity issue promptly.

Your output should be:

{
  "category": "Technical Issues",
"tags": ["internet connectivity", "slowdown", "frequent disconnections"]
"priority": "High"
}
```

```
Do not include any other text in the output except the JSON.

<</SYS>>[/INST]
"""
```

In [ ]:

```
#  create a new column llama_response'
# by applying the generate_llama_response function to each ticket in the 'support_ticket_
text' column of the DataFrame 'data_4'
data_4['llama_response'] = data_4['support_ticket_text'].apply(lambda x: generate_llama_
response(instruction_4, x))
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      28.29 ms /    48 runs   (    0.59 ms per tok
en,  1696.53 tokens per second)
llama_print_timings: prompt eval time =     915.81 ms /   494 tokens (    1.85 ms per tok
en,   539.41 tokens per second)
llama_print_timings:        eval time =    2554.20 ms /    47 runs   (   54.34 ms per tok
en,    18.40 tokens per second)
llama_print_timings:       total time =    3693.53 ms /   541 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      24.13 ms /    43 runs   (    0.56 ms per tok
en,  1782.01 tokens per second)
llama_print_timings: prompt eval time =     867.16 ms /   493 tokens (    1.76 ms per tok
en,   568.52 tokens per second)
llama_print_timings:        eval time =    2319.18 ms /    42 runs   (   55.22 ms per tok
en,    18.11 tokens per second)
llama_print_timings:       total time =    3371.22 ms /   535 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      21.92 ms /    41 runs   (    0.53 ms per tok
en,  1870.52 tokens per second)
llama_print_timings: prompt eval time =     868.02 ms /   489 tokens (    1.78 ms per tok
en,   563.35 tokens per second)
llama_print_timings:        eval time =    2252.10 ms /    40 runs   (   56.30 ms per tok
en,    17.76 tokens per second)
llama_print_timings:       total time =    3261.61 ms /   529 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      23.63 ms /    45 runs   (    0.53 ms per tok
en,  1904.36 tokens per second)
llama_print_timings: prompt eval time =     885.39 ms /   497 tokens (    1.78 ms per tok
en,   561.34 tokens per second)
llama_print_timings:        eval time =    2476.19 ms /    44 runs   (   56.28 ms per tok
en,    17.77 tokens per second)
llama_print_timings:       total time =    3525.01 ms /   541 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      26.87 ms /    45 runs   (    0.60 ms per tok
en,  1675.04 tokens per second)
llama_print_timings: prompt eval time =     852.49 ms /   472 tokens (    1.81 ms per tok
en,   553.67 tokens per second)
llama_print_timings:        eval time =    2438.55 ms /    44 runs   (   55.42 ms per tok
en,    18.04 tokens per second)
llama_print_timings:       total time =    3516.88 ms /   516 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      21.60 ms /    40 runs   (    0.54 ms per tok
en,  1851.85 tokens per second)
llama_print_timings: prompt eval time =     847.64 ms /   472 tokens (    1.80 ms per tok
en,   556.84 tokens per second)
llama_print_timings:        eval time =    2154.63 ms /    39 runs   (   55.25 ms per tok
en,    18.10 tokens per second)
llama_print_timings:       total time =    3158.59 ms /   511 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      24.14 ms /    47 runs   (    0.51 ms per tok
en,  1947.22 tokens per second)
llama_print_timings: prompt eval time =     857.00 ms /   475 tokens (    1.80 ms per tok
en,   554.26 tokens per second)
llama_print_timings:        eval time =    2614.13 ms /    46 runs   (   56.83 ms per tok
en,    17.60 tokens per second)
llama_print_timings:       total time =    3623.77 ms /   521 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      21.50 ms /    41 runs   (    0.52 ms per tok
en,  1907.15 tokens per second)
llama_print_timings: prompt eval time =     867.19 ms /   476 tokens (    1.82 ms per tok
en,   548.90 tokens per second)
llama_print_timings:        eval time =    2295.11 ms /    40 runs   (   57.38 ms per tok
en,    17.43 tokens per second)
llama_print_timings:       total time =    3308.51 ms /   516 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      26.06 ms /    43 runs   (    0.61 ms per tok
en,  1649.85 tokens per second)
llama_print_timings: prompt eval time =     860.26 ms /   470 tokens (    1.83 ms per tok
en,   546.35 tokens per second)
llama_print_timings:        eval time =    2330.55 ms /    42 runs   (   55.49 ms per tok
en,    18.02 tokens per second)
llama_print_timings:       total time =    3428.25 ms /   512 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      21.78 ms /    42 runs   (    0.52 ms per tok
en,  1928.82 tokens per second)
llama_print_timings: prompt eval time =     901.90 ms /   479 tokens (    1.88 ms per tok
en,   531.10 tokens per second)
llama_print_timings:        eval time =    2332.41 ms /    41 runs   (   56.89 ms per tok
en,    17.58 tokens per second)
llama_print_timings:       total time =    3379.35 ms /   520 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      22.59 ms /    44 runs   (    0.51 ms per tok
en,  1948.20 tokens per second)
llama_print_timings: prompt eval time =     876.88 ms /   472 tokens (    1.86 ms per tok
en,   538.27 tokens per second)
llama_print_timings:        eval time =    2462.58 ms /    43 runs   (   57.27 ms per tok
en,    17.46 tokens per second)
llama_print_timings:       total time =    3488.58 ms /   515 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      21.61 ms /    41 runs   (    0.53 ms per tok
en,  1897.36 tokens per second)
llama_print_timings: prompt eval time =     879.89 ms /   476 tokens (    1.85 ms per tok
en,   540.98 tokens per second)
llama_print_timings:        eval time =    2306.05 ms /    40 runs   (   57.65 ms per tok
en,    17.35 tokens per second)
llama_print_timings:       total time =    3327.06 ms /   516 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      26.00 ms /    42 runs   (    0.62 ms per tok
en,  1615.57 tokens per second)
llama_print_timings: prompt eval time =     880.01 ms /   476 tokens (    1.85 ms per tok
en,   540.90 tokens per second)
llama_print_timings:        eval time =    2270.19 ms /    41 runs   (   55.37 ms per tok
en,    18.06 tokens per second)
llama_print_timings:       total time =    3380.75 ms /   517 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      23.60 ms /    46 runs   (    0.51 ms per tok
en,  1948.99 tokens per second)
llama_print_timings: prompt eval time =     889.09 ms /   468 tokens (    1.90 ms per tok
en,   526.38 tokens per second)
llama_print_timings:        eval time =    2592.48 ms /    45 runs   (   57.61 ms per tok
en,    17.36 tokens per second)
llama_print_timings:       total time =    3631.39 ms /   513 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      20.10 ms /    39 runs   (    0.52 ms per tok
en,  1940.40 tokens per second)
llama_print_timings: prompt eval time =     891.65 ms /   473 tokens (    1.89 ms per tok
en,   530.47 tokens per second)
llama_print_timings:        eval time =    2187.59 ms /    38 runs   (   57.57 ms per tok
en,    17.37 tokens per second)
llama_print_timings:       total time =    3218.96 ms /   511 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      36.73 ms /    48 runs   (    0.77 ms per tok
en,  1306.87 tokens per second)
llama_print_timings: prompt eval time =     890.91 ms /   464 tokens (    1.92 ms per tok
en,   520.81 tokens per second)
llama_print_timings:        eval time =    2709.03 ms /    47 runs   (   57.64 ms per tok
en,    17.35 tokens per second)
llama_print_timings:       total time =    3841.15 ms /   511 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      28.05 ms /    47 runs   (    0.60 ms per tok
en,  1675.88 tokens per second)
llama_print_timings: prompt eval time =     896.27 ms /   469 tokens (    1.91 ms per tok
en,   523.28 tokens per second)
llama_print_timings:        eval time =    2621.12 ms /    46 runs   (   56.98 ms per tok
en,    17.55 tokens per second)
llama_print_timings:       total time =    3739.62 ms /   515 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      22.73 ms /    44 runs   (    0.52 ms per tok
en,  1935.60 tokens per second)
llama_print_timings: prompt eval time =     904.94 ms /   464 tokens (    1.95 ms per tok
en,   512.74 tokens per second)
llama_print_timings:        eval time =    2493.01 ms /    43 runs   (   57.98 ms per tok
en,    17.25 tokens per second)
llama_print_timings:       total time =    3549.39 ms /   507 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      22.42 ms /    44 runs   (    0.51 ms per tok
en,  1962.62 tokens per second)
llama_print_timings: prompt eval time =     922.76 ms /   465 tokens (    1.98 ms per tok
en,   503.92 tokens per second)
llama_print_timings:        eval time =    2548.18 ms /    43 runs   (   59.26 ms per tok
en,    16.87 tokens per second)
llama_print_timings:       total time =    3615.80 ms /   508 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      29.68 ms /    48 runs   (    0.62 ms per tok
en,  1617.47 tokens per second)
llama_print_timings: prompt eval time =    1466.98 ms /   621 tokens (    2.36 ms per tok
en,   423.32 tokens per second)
llama_print_timings:        eval time =    2772.87 ms /    47 runs   (   59.00 ms per tok
en,    16.95 tokens per second)
llama_print_timings:       total time =    4498.90 ms /   668 tokens
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      23.51 ms /    46 runs   (    0.51 ms per tok
en,  1956.28 tokens per second)
llama_print_timings: prompt eval time =    1649.14 ms /   734 tokens (    2.25 ms per tok
en,   445.08 tokens per second)
llama_print_timings:        eval time =    2921.46 ms /    45 runs   (   64.92 ms per tok
en,    15.40 tokens per second)
llama_print_timings:       total time =    4738.93 ms /   779 tokens
```

In [ ]:

```
# check the first five rows of the data to confirm whether the new column has been added
data_4['llama_response'].head()
```

Out[ ]:

```
0     {\n       "category": "Technical Issues",\n  ...
1     {\n       "category": "Hardware Issues",\n   ...
2     {\n       "category": "Data Recovery",\n     ...
3     {\n       "category": "Technical Issues",\n  ...
4     {\n       "category": "Hardware Issues",\n   ...
Name: llama_response, dtype: object
```

In [ ]:

```
# apply the extract_json_data function on the llama_response column to create a new colum
n called llama_response_parsed
data_4['llama_response_parsed'] = data_4['llama_response'].apply(extract_json_data)
data_4['llama_response_parsed'].head()
```

Out[ ]:

```
0    {'category': 'Technical Issues', 'tags': ['int...
1    {'category': 'Hardware Issues', 'tags': ['lapt...
2    {'category': 'Data Recovery', 'tags': ['delete...
3    {'category': 'Technical Issues', 'tags': ['wea...
4    {'category': 'Hardware Issues', 'tags': ['batt...
Name: llama_response_parsed, dtype: object
```

In [ ]:

```
# apply the json_normalize on llama_response_parsed variable
llama_response_parsed_df_4 = pd.json_normalize(data_4['llama_response_parsed'])
llama_response_parsed_df_4.head()
```

Out[ ]:

	category	tags	priority
0	Technical Issues	[internet connectivity, slowdown, frequent dis...	High
1	Hardware Issues	[laptop, startup, restart]	High
2	Data Recovery	[deleted documents, substantial data loss]	High
3	Technical Issues	[weak Wi-Fi signal, proximity to router]	Normal
4	Hardware Issues	[battery drain, rapidly draining]	Normal

In [ ]:

```
# concat data_4 and llama_response_parsed_df_4
data_with_parsed_model_output_4 = pd.concat([data_4, llama_response_parsed_df_4], axis=1
)
data_with_parsed_model_output_4.head()
```

Out[ ]:

support_tick_id	support_ticket_text	llama_response	llama_response_parsed	category	tags	priority
	My internet	{\n "category":			[internet	

	support_tick_id	support_ticket_text	llama_response	llama_response_parsed	category	tags	priority
0	ST2023-006	connection has significantly slowe...	"Technical Issues",\n ...	{'category': 'Technical Issues', 'tags': ['int...	Technical Issues	connectivity, slowdown, frequent dis...	High
1	ST2023-007	Urgent help required! My laptop refuses to sta...	{\n "category": "Hardware Issues",\n ...	{'category': 'Hardware Issues', 'tags': ['lapt...	Hardware Issues	[laptop, startup, restart]	High
2	ST2023-008	I've accidentally deleted essential work docum...	{\n "category": "Data Recovery",\n ...	{'category': 'Data Recovery', 'tags': ['delete...	Data Recovery	[deleted documents, substantial data loss]	High
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	{\n "category": "Technical Issues",\n ...	{'category': 'Technical Issues', 'tags': ['wea...	Technical Issues	[weak Wi-Fi signal, proximity to router]	Normal
4	ST2023-010	My smartphone battery is draining rapidly, eve...	{\n "category": "Hardware Issues",\n ...	{'category': 'Hardware Issues', 'tags': ['batt...	Hardware Issues	[battery drain, rapidly draining]	Normal

In [ ]:

```
# drop llama_response and llama_response_parsed variables
final_data_4 = data_with_parsed_model_output_4.drop(["llama_response","llama_response_par
sed"], axis=1)
final_data_4.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	category	tags	priority
0	ST2023-006	My internet connection has significantly slowe...	Technical Issues	[internet connectivity, slowdown, frequent dis...	High
1	ST2023-007	Urgent help required! My laptop refuses to sta...	Hardware Issues	[laptop, startup, restart]	High
2	ST2023-008	I've accidentally deleted essential work docum...	Data Recovery	[deleted documents, substantial data loss]	High
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	Technical Issues	[weak Wi-Fi signal, proximity to router]	Normal
4	ST2023-010	My smartphone battery is draining rapidly, eve...	Hardware Issues	[battery drain, rapidly draining]	Normal

Task 5 - Ticket Categorization, Creating Tags, Assigning Priority, Assigning ETA, and Returning Structured Output

In [ ]:

```
# create a copy of the data
data_5 = data.copy()
```

In [ ]:

```
## prompt to get the desired output
instruction_5 = """

    [INST]<<SYS>>
You are acting as a guide for a technical assistant. As a technical assistant, your prima
ry task is to classify the support ticket into specific categories. There are three categ
ories:

Technical Issues
Hardware Issues
Data Recovery

As a technical assistant, you should only respond with one of the predefined categories (
Technical Issues, Hardware Issues, and Data Recovery). Other responses are not acceptable
.
```

```
Your goal is to identify the category and then provide the following information from the
support ticket text, including the creating tags, assigning priority, assigning ETA. Go t
hrough each support ticket text thoroughly and considering the overall sentiment before r
esponding, and generate only a structured output for the JSON format.

Here is the structured output for further analysis in JSON format. Follow the JSON format
below strictly. You must include the following as JSON output:

{"category": <create one of the predefined categories (Technical Issues, Hardware Issues,
and Data Recovery). Other responses are not acceptable>,
"tags": <create tags to further classify the ticket>,
"priority": <assign a priority level (e.g., "High" or "Normal") only based on the underst
anding of the text>,
"ETA": <provide an estimated time for resolving the issue mentioned in the ticket>
}

Here is an example:

support_ticket_text: My internet connection has significantly slowed down over the past t
wo days, making it challenging to work efficiently from home. Frequent disconnections are
causing major disruptions. Please assist in resolving this connectivity issue promptly.

Your output should be:

{
  "category": "Technical Issues",
  "tags": ["internet connectivity", "slowdown", "frequent disconnections"],
  "priority": "High",
  "ETA": "ASAP"
}



Do not include any other text in the output except the JSON.

<</SYS>>[/INST]
"""
```

In [ ]:

```
# complete the code to create a new column llama_response'
# by applying the generate_llama_response function to each ticket in the 'support_ticket_
text' column of the DataFrame 'data_5'
data_5['llama_response'] = data_5['support_ticket_text'].apply(lambda x: generate_llama_
response(instruction_5, x))
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      29.95 ms /    58 runs   (    0.52 ms per tok
en,  1936.82 tokens per second)
llama_print_timings: prompt eval time =    1435.30 ms /   531 tokens (    2.70 ms per tok
en,   369.96 tokens per second)
llama_print_timings:        eval time =    3170.15 ms /    57 runs   (   55.62 ms per tok
en,    17.98 tokens per second)
llama_print_timings:       total time =    4804.18 ms /   588 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      32.28 ms /    54 runs   (    0.60 ms per tok
en,  1672.71 tokens per second)
llama_print_timings: prompt eval time =    1378.02 ms /   530 tokens (    2.60 ms per tok
en,   384.61 tokens per second)
llama_print_timings:        eval time =    2933.51 ms /    53 runs   (   55.35 ms per tok
en,    18.07 tokens per second)
llama_print_timings:       total time =    4595.58 ms /   583 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      28.41 ms /    51 runs   (    0.56 ms per tok
en,  1794.95 tokens per second)
llama_print_timings: prompt eval time =    1327.79 ms /   526 tokens (    2.52 ms per tok
en,   396.15 tokens per second)
llama_print_timings:        eval time =    2849.24 ms /    50 runs   (   56.98 ms per tok
en,    17.55 tokens per second)
llama_print_timings:       total time =    4356.35 ms /   576 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      27.86 ms /    55 runs   (    0.51 ms per tok
en,  1974.51 tokens per second)
llama_print_timings: prompt eval time =    1492.89 ms /   534 tokens (    2.80 ms per tok
en,   357.69 tokens per second)
llama_print_timings:        eval time =    3104.54 ms /    54 runs   (   57.49 ms per tok
en,    17.39 tokens per second)
llama_print_timings:       total time =    4789.76 ms /   588 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      34.85 ms /    59 runs   (    0.59 ms per tok
en,  1692.82 tokens per second)
llama_print_timings: prompt eval time =     939.27 ms /   509 tokens (    1.85 ms per tok
en,   541.91 tokens per second)
llama_print_timings:        eval time =    3242.70 ms /    58 runs   (   55.91 ms per tok
en,    17.89 tokens per second)
llama_print_timings:       total time =    4483.24 ms /   567 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      25.32 ms /    50 runs   (    0.51 ms per tok
en,  1974.65 tokens per second)
llama_print_timings: prompt eval time =     952.20 ms /   509 tokens (    1.87 ms per tok
en,   534.55 tokens per second)
llama_print_timings:        eval time =    2844.21 ms /    49 runs   (   58.05 ms per tok
en,    17.23 tokens per second)
llama_print_timings:       total time =    3965.38 ms /   558 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      28.60 ms /    55 runs   (    0.52 ms per tok
en,  1922.81 tokens per second)
llama_print_timings: prompt eval time =     986.96 ms /   512 tokens (    1.93 ms per tok
en,   518.77 tokens per second)
llama_print_timings:        eval time =    3128.14 ms /    54 runs   (   57.93 ms per tok
en,    17.26 tokens per second)
llama_print_timings:       total time =    4319.10 ms /   566 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      31.76 ms /    51 runs   (    0.62 ms per tok
en,  1605.84 tokens per second)
llama_print_timings: prompt eval time =    1005.36 ms /   512 tokens (    1.96 ms per tok
en,   509.27 tokens per second)
llama_print_timings:        eval time =    2828.04 ms /    51 runs   (   55.45 ms per tok
en,    18.03 tokens per second)
llama_print_timings:       total time =    4116.29 ms /   563 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      27.32 ms /    53 runs   (    0.52 ms per tok
en,  1940.11 tokens per second)
llama_print_timings: prompt eval time =     981.63 ms /   507 tokens (    1.94 ms per tok
en,   516.49 tokens per second)
llama_print_timings:        eval time =    3152.64 ms /    52 runs   (   60.63 ms per tok
en,    16.49 tokens per second)
llama_print_timings:       total time =    4321.54 ms /   559 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      26.13 ms /    52 runs   (    0.50 ms per tok
en,  1989.75 tokens per second)
llama_print_timings: prompt eval time =    1125.90 ms /   516 tokens (    2.18 ms per tok
en,   458.30 tokens per second)
llama_print_timings:        eval time =    3070.98 ms /    51 runs   (   60.22 ms per tok
en,    16.61 tokens per second)
llama_print_timings:       total time =    4376.30 ms /   567 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      35.73 ms /    54 runs   (    0.66 ms per tok
en,  1511.25 tokens per second)
llama_print_timings: prompt eval time =     989.30 ms /   509 tokens (    1.94 ms per tok
en,   514.51 tokens per second)
llama_print_timings:        eval time =    2938.76 ms /    53 runs   (   55.45 ms per tok
en,    18.03 tokens per second)
llama_print_timings:       total time =    4213.93 ms /   562 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      26.25 ms /    51 runs   (    0.51 ms per tok
en,  1943.08 tokens per second)
llama_print_timings: prompt eval time =    1020.65 ms /   512 tokens (    1.99 ms per tok
en,   501.64 tokens per second)
llama_print_timings:        eval time =    3103.61 ms /    51 runs   (   60.86 ms per tok
en,    16.43 tokens per second)
llama_print_timings:       total time =    4308.20 ms /   563 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      26.45 ms /    52 runs   (    0.51 ms per tok
en,  1965.97 tokens per second)
llama_print_timings: prompt eval time =    1022.96 ms /   512 tokens (    2.00 ms per tok
en,   500.51 tokens per second)
llama_print_timings:        eval time =    3247.75 ms /    52 runs   (   62.46 ms per tok
en,    16.01 tokens per second)
llama_print_timings:       total time =    4449.51 ms /   564 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      33.42 ms /    56 runs   (    0.60 ms per tok
en,  1675.59 tokens per second)
llama_print_timings: prompt eval time =     995.54 ms /   505 tokens (    1.97 ms per tok
en,   507.26 tokens per second)
llama_print_timings:        eval time =    3228.44 ms /    55 runs   (   58.70 ms per tok
en,    17.04 tokens per second)
llama_print_timings:       total time =    4521.14 ms /   560 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      27.06 ms /    53 runs   (    0.51 ms per tok
en,  1958.90 tokens per second)
llama_print_timings: prompt eval time =    1010.04 ms /   510 tokens (    1.98 ms per tok
en,   504.93 tokens per second)
llama_print_timings:        eval time =    3265.95 ms /    52 runs   (   62.81 ms per tok
en,    15.92 tokens per second)
llama_print_timings:       total time =    4463.69 ms /   562 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      29.87 ms /    58 runs   (    0.52 ms per tok
en,  1941.55 tokens per second)
llama_print_timings: prompt eval time =    1005.12 ms /   501 tokens (    2.01 ms per tok
en,   498.45 tokens per second)
llama_print_timings:        eval time =    3622.38 ms /    57 runs   (   63.55 ms per tok
en,    15.74 tokens per second)
llama_print_timings:       total time =    4819.94 ms /   558 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      33.88 ms /    57 runs   (    0.59 ms per tok
en,  1682.56 tokens per second)
llama_print_timings: prompt eval time =    1011.35 ms /   506 tokens (    2.00 ms per tok
en,   500.32 tokens per second)
llama_print_timings:        eval time =    3420.68 ms /    56 runs   (   61.08 ms per tok
en,    16.37 tokens per second)
llama_print_timings:       total time =    4712.53 ms /   562 tokens
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      26.50 ms /    52 runs   (    0.51 ms per tok
en,  1962.56 tokens per second)
llama_print_timings: prompt eval time =    1014.95 ms /   501 tokens (    2.03 ms per tok
en,   493.62 tokens per second)
llama_print_timings:        eval time =    3302.97 ms /    51 runs   (   64.76 ms per tok
en,    15.44 tokens per second)
llama_print_timings:       total time =    4500.76 ms /   552 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      27.43 ms /    54 runs   (    0.51 ms per tok
en,  1969.01 tokens per second)
llama_print_timings: prompt eval time =    1018.27 ms /   502 tokens (    2.03 ms per tok
en,   492.99 tokens per second)
llama_print_timings:        eval time =    3433.18 ms /    53 runs   (   64.78 ms per tok
en,    15.44 tokens per second)
llama_print_timings:       total time =    4639.94 ms /   555 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      31.75 ms /    58 runs   (    0.55 ms per tok
en,  1826.66 tokens per second)
llama_print_timings: prompt eval time =    1631.58 ms /   658 tokens (    2.48 ms per tok
en,   403.29 tokens per second)
llama_print_timings:        eval time =    3672.66 ms /    57 runs   (   64.43 ms per tok
en,    15.52 tokens per second)
llama_print_timings:       total time =    5551.23 ms /   715 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      25.87 ms /    51 runs   (    0.51 ms per tok
en,  1971.47 tokens per second)
llama_print_timings: prompt eval time =    1854.72 ms /   771 tokens (    2.41 ms per tok
en,   415.70 tokens per second)
llama_print_timings:        eval time =    3361.68 ms /    50 runs   (   67.23 ms per tok
en,    14.87 tokens per second)
llama_print_timings:       total time =    5404.58 ms /   821 tokens
```

In [ ]:

```
# check the first five rows of the data to confirm whether the new column has been added
data_5['llama_response'].head()
```

Out[ ]:

```
0     {\n       "category": "Technical Issues",\n  ...
1     {\n       "category": "Hardware Issues",\n   ...
2     {\n       "category": "Data Recovery",\n     ...
3     {\n       "category": "Hardware Issues",\n   ...
4     {\n       "category": "Hardware Issues",\n   ...
Name: llama_response, dtype: object
```

In [ ]:

```
## apply the extract_json_data function on the llama_response column to create a new colu
mn called llama_response_parsed
data_5['llama_response_parsed'] = data_5['llama_response'].apply(extract_json_data)
data_5['llama_response_parsed'].head()
```

Out[ ]:

```
0    {'category': 'Technical Issues', 'tags': ['int...
1    {'category': 'Hardware Issues', 'tags': ['lapt...
2    {'category': 'Data Recovery', 'tags': ['delete...
3    {'category': 'Hardware Issues', 'tags': ['weak...
4    {'category': 'Hardware Issues', 'tags': ['batt...
Name: llama_response_parsed, dtype: object
```

In [ ]:

```python
# apply the json_normalize on llama_response_parsed variable
llama_response_parsed_df_5 = pd.json_normalize(data_5['llama_response_parsed'])
llama_response_parsed_df_5.head()
```

Out[ ]:

	category	tags	priority	ETA
0	Technical Issues	[internet connectivity, slowdown, frequent dis...	High	ASAP
1	Hardware Issues	[laptop, startup, restart]	High	24 hours
2	Data Recovery	[deleted documents, substantial data loss]	High	ASAP
3	Hardware Issues	[weak signal strength, persistent issue]	Normal	Within 24 hours
4	Hardware Issues	[battery drain, rapidly draining]	Normal	Within 24 hours

In [ ]:

```python
# concat data_5 and llama_response_parsed_df_5
data_with_parsed_model_output_5 = pd.concat([data_5, llama_response_parsed_df_5], axis=1
)
data_with_parsed_model_output_5.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	llama_response	llama_response_parsed	category	tags	priority	ETA
0	ST2023-006	My internet connection has significantly slowe...	{\n "category": "Technical Issues",\n ...	{'category': 'Technical Issues', 'tags': ['int...	Technical Issues	[internet connectivity, slowdown, frequent dis...	High	ASAP
1	ST2023-007	Urgent help required! My laptop refuses to sta...	{\n "category": "Hardware Issues",\n ...	{'category': 'Hardware Issues', 'tags': ['lapt...	Hardware Issues	[laptop, startup, restart]	High	24 hours
2	ST2023-008	I've accidentally deleted essential work docum...	{\n "category": "Data Recovery",\n ...	{'category': 'Data Recovery', 'tags': ['delete...	Data Recovery	[deleted documents, substantial data loss]	High	ASAP
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	{\n "category": "Hardware Issues",\n ...	{'category': 'Hardware Issues', 'tags': ['weak...	Hardware Issues	[weak signal strength, persistent issue]	Normal	Within 24 hours
4	ST2023-010	My smartphone battery is draining rapidly, eve...	{\n "category": "Hardware Issues",\n ...	{'category': 'Hardware Issues', 'tags': ['batt...	Hardware Issues	[battery drain, rapidly draining]	Normal	Within 24 hours

In [ ]:

```python
## drop llama_response and llama_response_parsed variables
final_data_5 = data_with_parsed_model_output_5.drop(['llama_response','llama_response_par
sed'], axis=1)
final_data_5.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	category	tags	priority	ETA
0	ST2023-006	My internet connection has significantly slowe...	Technical Issues	[internet connectivity, slowdown, frequent dis...	High	ASAP
1	ST2023-007	Urgent help required! My laptop refuses to sta...	Hardware Issues	[laptop, startup, restart]	High	24 hours
2	ST2023-008	I've accidentally deleted essential work docum...	Data Recovery	[deleted documents, substantial data loss]	High	ASAP
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	Hardware Issues	[weak signal strength, persistent issue]	Normal	Within 24 hours
4	ST2023-010	My smartphone battery is draining	Hardware	[battery drain, rapidly draining]	Normal	Within 24

4	ST2023-010	rapidly, eye...	Issues	[battery drain, rapidly draining]	Normal	hours
	support_tick_id	support_ticket_text	category	tags	priority	ETA

Task 6 - Ticket Categorization, Creating Tags, Assigning Priority, Assigning ETA, Creating a Draft Response, and Returning Structured Output

In [ ]:

```
# create a copy of the data
data_6 = data.copy()
```

In [ ]:

```
## prompt to get the desired output
instruction_6 = """
    [INST]<<SYS>> You are acting as a guide for a technical assistant. As a technical ass
istant, your primary task is to classify the support ticket into specific categories. The
re are three categories:
    Technical Issues
    Hardware Issues
    Data Recovery.

    As a technical assistant, you should only respond with one of the predefined categori
es (Technical Issues, Hardware Issues, and Data Recovery). Other responses are not accept
able.

    Your goal is to identify the category and then provide the following information from
the support ticket text, including the creating tags, assigning priority, assigning ETA,
generating a response based on sentiment. Go through the each support ticket text thoroug
hly and considering the overall sentiment before responding, and generate only a structur
ed output for the JSON format.
    Here is the structured output for further analysis in JSON format. Follow the JSON fo
rmat below strictly.You must include the following as JSON output:


    {"category": <create one of the predefined categories (Technical Issues, Hardware Iss
ues, and Data Recovery). Other responses are not acceptable>,
     "tags": <create tags to further classify the ticket>,
     "priority": <assign a priority level (e.g., "High" or "Normal") only based on the un
derstanding of the text>,
     "ETA": <provide an estimated time for resolving the issue mentioned in the ticket>,
     "response": <generate a 50 words reply that aligns with the sentiment expressed in t
he ticket.The tone of all responses should be polite and professional.>
     }

     Here is an example:

     support_ticket_text: My internet connection has significantly slowed down over the p
ast two days, making it challenging to work efficiently from home. Frequent disconnection
s are causing major disruptions. Please assist in resolving this connectivity issue promp
tly.

     Your output should be:

     {
      "category": "Technical Issues",
      "tags": ["internet connectivity", "slowdown", "frequent disconnections"],
      "priority": "High",
      "ETA": "ASAP",
      "response": "We are sorry to hear the slow internet connectivity issues. Our team w
ill work promptly to resolve this issue and get your connection back up to speed."
      }

      Do not include any other text in the output except the JSON.



    <</SYS>>[/INST]
    """
```

In [ ]:

In [ ]:

```
# create a new column llama_response'
# by applying the generate_llama_response function to each ticket in the 'support_ticket_
text' column of the DataFrame 'data_6'
data_6['llama_response'] = data_6['support_ticket_text'].apply(lambda x: generate_llama_
response(instruction_6, x))
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    968.44 ms
llama_print_timings:      sample time =     52.58 ms /    90 runs   (    0.58 ms per tok
en,  1711.84 tokens per second)
llama_print_timings: prompt eval time =   1387.82 ms /   628 tokens (    2.21 ms per tok
en,   452.51 tokens per second)
llama_print_timings:        eval time =   4678.44 ms /    89 runs   (   52.57 ms per tok
en,    19.02 tokens per second)
llama_print_timings:       total time =   6497.33 ms /   717 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    968.44 ms
llama_print_timings:      sample time =     59.88 ms /   112 runs   (    0.53 ms per tok
en,  1870.53 tokens per second)
llama_print_timings: prompt eval time =   1321.31 ms /   627 tokens (    2.11 ms per tok
en,   474.53 tokens per second)
llama_print_timings:        eval time =   6178.66 ms /   111 runs   (   55.66 ms per tok
en,    17.97 tokens per second)
llama_print_timings:       total time =   7909.22 ms /   738 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    968.44 ms
llama_print_timings:      sample time =     50.95 ms /    89 runs   (    0.57 ms per tok
en,  1746.74 tokens per second)
llama_print_timings: prompt eval time =   1329.15 ms /   623 tokens (    2.13 ms per tok
en,   468.72 tokens per second)
llama_print_timings:        eval time =   4827.29 ms /    88 runs   (   54.86 ms per tok
en,    18.23 tokens per second)
llama_print_timings:       total time =   6570.34 ms /   711 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    968.44 ms
llama_print_timings:      sample time =     48.41 ms /    94 runs   (    0.51 ms per tok
en,  1941.79 tokens per second)
llama_print_timings: prompt eval time =   1351.35 ms /   631 tokens (    2.14 ms per tok
en,   466.94 tokens per second)
llama_print_timings:        eval time =   5306.75 ms /    93 runs   (   57.06 ms per tok
en,    17.52 tokens per second)
llama_print_timings:       total time =   6999.85 ms /   724 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    968.44 ms
llama_print_timings:      sample time =     66.19 ms /   110 runs   (    0.60 ms per tok
en,  1661.83 tokens per second)
llama_print_timings: prompt eval time =   1332.89 ms /   606 tokens (    2.20 ms per tok
en,   454.65 tokens per second)
llama_print_timings:        eval time =   6110.64 ms /   109 runs   (   56.06 ms per tok
en,    17.84 tokens per second)
llama_print_timings:       total time =   7961.09 ms /   715 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    968.44 ms
llama_print_timings:      sample time =     77.19 ms /    91 runs   (    0.85 ms per tok
en,  1178.91 tokens per second)
llama_print_timings: prompt eval time =   1345.31 ms /   606 tokens (    2.22 ms per tok
en,   450.45 tokens per second)
llama_print_timings:        eval time =   5089.88 ms /    90 runs   (   56.55 ms per tok
en,    17.68 tokens per second)
llama_print_timings:       total time =   6868.94 ms /   696 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    968.44 ms
llama_print_timings:      sample time =     58.63 ms /   102 runs   (    0.57 ms per tok
en,  1739.75 tokens per second)
```

```
llama_print_timings: prompt eval time =    1365.67 ms /   609 tokens (    2.24 ms per tok
en,   445.93 tokens per second)
llama_print_timings:        eval time =    5722.72 ms /   101 runs   (   56.66 ms per tok
en,    17.65 tokens per second)
llama_print_timings:       total time =    7609.77 ms /   710 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      55.01 ms /    99 runs   (    0.56 ms per tok
en,  1799.77 tokens per second)
llama_print_timings: prompt eval time =    1378.67 ms /   610 tokens (    2.26 ms per tok
en,   442.46 tokens per second)
llama_print_timings:        eval time =    5572.47 ms /    98 runs   (   56.86 ms per tok
en,    17.59 tokens per second)
llama_print_timings:       total time =    7356.85 ms /   708 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      46.85 ms /    90 runs   (    0.52 ms per tok
en,  1921.19 tokens per second)
llama_print_timings: prompt eval time =    1388.37 ms /   604 tokens (    2.30 ms per tok
en,   435.04 tokens per second)
llama_print_timings:        eval time =    5158.44 ms /    89 runs   (   57.96 ms per tok
en,    17.25 tokens per second)
llama_print_timings:       total time =    6877.06 ms /   693 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      52.55 ms /    93 runs   (    0.57 ms per tok
en,  1769.81 tokens per second)
llama_print_timings: prompt eval time =    1419.52 ms /   613 tokens (    2.32 ms per tok
en,   431.84 tokens per second)
llama_print_timings:        eval time =    5328.54 ms /    92 runs   (   57.92 ms per tok
en,    17.27 tokens per second)
llama_print_timings:       total time =    7159.12 ms /   705 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      78.62 ms /    93 runs   (    0.85 ms per tok
en,  1182.84 tokens per second)
llama_print_timings: prompt eval time =    1433.26 ms /   606 tokens (    2.37 ms per tok
en,   422.81 tokens per second)
llama_print_timings:        eval time =    5372.47 ms /    92 runs   (   58.40 ms per tok
en,    17.12 tokens per second)
llama_print_timings:       total time =    7377.05 ms /   698 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      49.73 ms /    83 runs   (    0.60 ms per tok
en,  1668.98 tokens per second)
llama_print_timings: prompt eval time =    1456.85 ms /   610 tokens (    2.39 ms per tok
en,   418.71 tokens per second)
llama_print_timings:        eval time =    4939.15 ms /    82 runs   (   60.23 ms per tok
en,    16.60 tokens per second)
llama_print_timings:       total time =    6797.21 ms /   692 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      50.87 ms /    98 runs   (    0.52 ms per tok
en,  1926.56 tokens per second)
llama_print_timings: prompt eval time =    1472.87 ms /   610 tokens (    2.41 ms per tok
en,   414.16 tokens per second)
llama_print_timings:        eval time =    6158.26 ms /    97 runs   (   63.49 ms per tok
en,    15.75 tokens per second)
llama_print_timings:       total time =    7982.30 ms /   707 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =     151.80 ms /    99 runs   (    1.53 ms per tok
en,   652.19 tokens per second)
llama_print_timings: prompt eval time =    1476.24 ms /   602 tokens (    2.45 ms per tok
en,   407.79 tokens per second)
```

```
llama_print_timings:        eval time =    6012.72 ms /    98 runs   (   61.35 ms per tok
en,    16.30 tokens per second)
llama_print_timings:       total time =    8356.06 ms /   700 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      51.76 ms /    91 runs   (    0.57 ms per tok
en,  1758.28 tokens per second)
llama_print_timings: prompt eval time =    1499.24 ms /   607 tokens (    2.47 ms per tok
en,   404.87 tokens per second)
llama_print_timings:        eval time =    5658.11 ms /    90 runs   (   62.87 ms per tok
en,    15.91 tokens per second)
llama_print_timings:       total time =    7619.15 ms /   697 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      50.77 ms /    90 runs   (    0.56 ms per tok
en,  1772.67 tokens per second)
llama_print_timings: prompt eval time =    1486.30 ms /   598 tokens (    2.49 ms per tok
en,   402.34 tokens per second)
llama_print_timings:        eval time =    5724.61 ms /    89 runs   (   64.32 ms per tok
en,    15.55 tokens per second)
llama_print_timings:       total time =    7583.36 ms /   687 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      54.85 ms /    97 runs   (    0.57 ms per tok
en,  1768.33 tokens per second)
llama_print_timings: prompt eval time =    1506.09 ms /   603 tokens (    2.50 ms per tok
en,   400.38 tokens per second)
llama_print_timings:        eval time =    6154.40 ms /    96 runs   (   64.11 ms per tok
en,    15.60 tokens per second)
llama_print_timings:       total time =    8108.30 ms /   699 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      52.41 ms /   102 runs   (    0.51 ms per tok
en,  1946.08 tokens per second)
llama_print_timings: prompt eval time =    1499.33 ms /   598 tokens (    2.51 ms per tok
en,   398.85 tokens per second)
llama_print_timings:        eval time =    6644.53 ms /   101 runs   (   65.79 ms per tok
en,    15.20 tokens per second)
llama_print_timings:       total time =    8509.20 ms /   699 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      51.64 ms /    88 runs   (    0.59 ms per tok
en,  1704.07 tokens per second)
llama_print_timings: prompt eval time =    1521.04 ms /   599 tokens (    2.54 ms per tok
en,   393.81 tokens per second)
llama_print_timings:        eval time =    5498.77 ms /    87 runs   (   63.20 ms per tok
en,    15.82 tokens per second)
llama_print_timings:       total time =    7445.60 ms /   686 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      48.02 ms /    90 runs   (    0.53 ms per tok
en,  1874.26 tokens per second)
llama_print_timings: prompt eval time =    1716.97 ms /   755 tokens (    2.27 ms per tok
en,   439.73 tokens per second)
llama_print_timings:        eval time =    5880.26 ms /    89 runs   (   66.07 ms per tok
en,    15.14 tokens per second)
llama_print_timings:       total time =    7939.94 ms /   844 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =     968.44 ms
llama_print_timings:      sample time =      50.84 ms /    94 runs   (    0.54 ms per tok
en,  1849.05 tokens per second)
llama_print_timings: prompt eval time =    1919.33 ms /   868 tokens (    2.21 ms per tok
en,   452.24 tokens per second)
llama_print_timings:        eval time =    6168.78 ms /    93 runs   (   66.33 ms per tok
en,    15.08 tokens per second)
```

```
llama_print_timings:       total time =    8444.51 ms /   961 tokens
```

In [ ]:

```
#  check the first five rows of the data to confirm whether the new column has been added
data_6['llama_response'].head()
```

Out[ ]:

```
0    \n{\n"category": "Technical Issues",\n"tags": ...
1    \n{\n"category": "Hardware Issues",\n"tags": [...
2    \n    {\n       "category": "Data Recovery",\n ...
3    \n    {\n       "category": "Hardware Issues",\...
4    \n     {\n      "category": "Hardware Issues",\...
Name: llama_response, dtype: object
```

In [ ]:

```
# apply the extract_json_data function on the llama_response column to create a new colum
n called llama_response_parsed
data_6['llama_response_parsed'] = data_6['llama_response'].apply(extract_json_data)
data_6['llama_response_parsed'].head()
```

Out[ ]:

```
0    {'category': 'Technical Issues', 'tags': ['int...
1    {'category': 'Hardware Issues', 'tags': ['lapt...
2    {'category': 'Data Recovery', 'tags': ['delete...
3    {'category': 'Hardware Issues', 'tags': ['weak...
4    {'category': 'Hardware Issues', 'tags': ['batt...
Name: llama_response_parsed, dtype: object
```

In [ ]:

```
# apply the normalize on llama_response_parsed variable
llama_response_parsed_df_6= pd.json_normalize(data_6['llama_response_parsed'])
llama_response_parsed_df_6.head()
```

Out[ ]:

	category	tags	priority	ETA	response
0	Technical Issues	[internet connectivity, slowdown, frequent dis...	High	ASAP	We are sorry to hear the slow internet connect...
1	Hardware Issues	[laptop, startup, crucial presentation]	High	ASAP	We understand the urgency of your situation. B...
2	Data Recovery	[deleted documents, substantial data loss]	High	ASAP	We understand the urgency of recovering your d...
3	Hardware Issues	[weak signal strength, persistent issue]	Normal	24-48 hours	We understand your frustration with the weak W...
4	Hardware Issues	[battery drain, smartphone]	Normal	Shortly	Thank you for reaching out. We're here to help...

In [ ]:

```
# concat data_6 and llama_response_parsed_df_6
data_with_parsed_model_output_6 = pd.concat([data_6, llama_response_parsed_df_6], axis=1
)
data_with_parsed_model_output_6.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	llama_response	llama_response_parsed	category	tags	priority	ETA
0	ST2023-006	My internet connection has significantly slowe...	\n{\n"category": "Technical Issues",\n"tags": ...	{'category': 'Technical Issues', 'tags': ['int...	Technical Issues	[internet connectivity, slowdown, frequent dis...	High	ASAP

	support_tick_id	support_ticket_text	llama_response	llama_response_parsed	category	tags	priority	ETA	
1	ST2023-007	Urgent help required! My laptop refuses to sta...	\n{\n"category": "Hardware Issues",\n"tags": [...	{'category': 'Hardware Issues', 'tags': ['lapt...	Hardware Issues	[laptop, startup, crucial presentation]	High	ASAP	un u
2	ST2023-008	I've accidentally deleted essential work docum...	\n {\n "category": "Data Recovery",\n ...	{'category': 'Data Recovery', 'tags': ['delete...	Data Recovery	[deleted documents, substantial data loss]	High	ASAP	un u re
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	\n {\n "category": "Hardware Issues",\...	{'category': 'Hardware Issues', 'tags': ['weak...	Hardware Issues	[weak signal strength, persistent issue]	Normal	24-48 hours	un fı
4	ST2023-010	My smartphone battery is draining rapidly, eve...	\n {\n "category": "Hardware Issues",\...	{'category': 'Hardware Issues', 'tags': ['batt...	Hardware Issues	[battery drain, smartphone]	Normal	Shortly	T c

In [ ]:

```
# drop llama_response and llama_response_parsed variables
final_data_6 = data_with_parsed_model_output_6.drop(['llama_response','llama_response_par
sed'], axis=1)
final_data_6.head()
```

Out[ ]:

	support_tick_id	support_ticket_text	category	tags	priority	ETA	response
0	ST2023-006	My internet connection has significantly slowe...	Technical Issues	[internet connectivity, slowdown, frequent dis...	High	ASAP	We are sorry to hear the slow internet connect...
1	ST2023-007	Urgent help required! My laptop refuses to sta...	Hardware Issues	[laptop, startup, crucial presentation]	High	ASAP	We understand the urgency of your situation. B...
2	ST2023-008	I've accidentally deleted essential work docum...	Data Recovery	[deleted documents, substantial data loss]	High	ASAP	We understand the urgency of recovering your d...
3	ST2023-009	Despite being in close proximity to my Wi-Fi r...	Hardware Issues	[weak signal strength, persistent issue]	Normal	24-48 hours	We understand your frustration with the weak W...
4	ST2023-010	My smartphone battery is draining rapidly, eve...	Hardware Issues	[battery drain, smartphone]	Normal	Shortly	Thank you for reaching out. We're here to help...

Model Output Analysis

Univariate analysis of the columns in the final structured dataframe obtained from the model's output.

In [ ]:

```
# creating a copy of the dataframe
final_data = final_data_6.copy()
```

In [ ]:

```
# check the distribution of categories
```

```
final_data['category'].value_counts()
```

Out[ ]:

```
category
Technical Issues    7
Hardware Issues     7
Data Recovery       7
Name: count, dtype: int64
```

Observation:

- **The data exhibit a uniform distribution within this sample, as all categories within the 'category' column ("Technical Issues", "Hardware Issues", "Data Recovery") possess an identical frequency of 7.**

In [ ]:

```
# check the distribution of priority
final_data['priority'].value_counts()
```

Out[ ]:

```
priority
High      17
Normal     4
Name: count, dtype: int64
```

Observation:

- **High priority issues are significantly more frequent than Normal priority issues**

In [ ]:

```
# check the distribution of ETA
final_data['ETA'].value_counts()
```

Out[ ]:

```
ETA
ASAP           17
Shortly         2
24-48 hours     1
24 hours        1
Name: count, dtype: int64
```

Observation:

- **Majority of cases require immediate attention ("ASAP"), with only a few spread across "Shortly", "24-48 hours" and "24 hours".**

In [ ]:

```
# check the distribution of priority by categories
final_data.groupby(['priority', 'category']).support_tick_id.count()
```

Out[ ]:

```
priority  category
High      Data Recovery       7
          Hardware Issues     4
          Technical Issues    6
Normal    Hardware Issues     3
          Technical Issues    1
Name: support_tick_id, dtype: int64
```

Observation

- **High priority is spread across all categories, whereas Normal priority is less frequent and mostly associated with Hardware and Technical Issues.**

In [ ]:

```
# check the distribution of ETA by categories
final_data.groupby(['ETA', 'category']).support_tick_id.count()
```

Out[ ]:

```
ETA           category
24 hours      Hardware Issues     1
24-48 hours   Hardware Issues     1
ASAP          Data Recovery       7
              Hardware Issues     4
              Technical Issues    6
Shortly       Hardware Issues     1
              Technical Issues    1
Name: support_tick_id, dtype: int64
```

Observation:

- **Most "ASAP" cases are spread across all categories, while other ETAs are less frequent and more category-specific.**

Actionable Insights and Recommendations

Share your observations and insights from this exercise, and your recommendations for a business looking to adopt a solution such as this

Observations and insights:

- **The project aims to create a Generative AI application to automate the processing and classification of support tickets using a Large Language Model. To achieve this, the Llama 2 model has been utilized. This model is a pre-trained and fine-tuned generative text model ranging from 7 billion to 70 billion parameters. Specifically, the 13B fine-tuned model (llama-2-13b-chat.Q6_K.gguf, a 6-bit quantized version model) optimized for dialogue use cases has been utilized, and it has been converted to the Hugging Face Transformers format.**
- **The code for LLaMA Response Generation with Instruction Integration has been created, effectively utilizing the LLaMA model to generate customized responses by integrating reviews with specific instructions. It ensures coherence and relevance through meticulous parameter selection. The modular structure enhances integration and maintenance ease. However, successful execution relies on the reliability and accessibility of the LLaMA model implementation. Overall, it presents a promising approach for generating responses across diverse applications.**
- **Next, a response text from the LLaMA model was generated using the lcpp_llm instance with the following parameters: max_tokens = 1024, temperature = 0.01, top_p = 0.95, repeat_penalty = 1.2, top_k = 50, and echo = False.**
- **Model response parameters (tasks) are being created with multiple instructions based on the provided tasks. These instructions contain detailed guidelines to help the technical assistant classify the support ticket text, generate tags, assign priority, suggest the expected time of arrival (ETA), and create a sentiment-based response in JSON format. A one-shot example has also been included to guide the technical assistant.**
- **For each task (1, 2, 3, 4, 5, 6), I parsed the JSON data, extracted key-value pairs, normalized, and concatenated them (setting axis = 1). Finally, I arrived at the final_data_6 DataFrame (from task_6) with JSON responses in five new columns: category, tags, priority, ETA, and response.**

Recommendations:

1. **Pilot Implementation and Proof of Concept: Before fully integrating the solution into your operational workflow, it's advisable to conduct a pilot implementation or proof of concept (POC). Start by selecting a subset of support tickets or a specific department to test the solution's effectiveness in real-world**

scenarios. This allows you to assess its performance, identify any potential challenges or limitations, and gather feedback from users. Based on the results of the pilot, you can make informed decisions about scaling up the implementation and fine-tuning the solution to better align with your organization's needs.
2. **User Training and Change Management:** Implementing an AI-driven support ticket categorization solution involves a shift in how support teams interact with and manage tickets. To ensure smooth adoption and maximize the benefits of the new system, invest in comprehensive user training and change management initiatives. Provide training sessions to familiarize support staff with the features and functionalities of the solution, including how to interpret the categorization results, prioritize tickets, and utilize any additional insights provided by the system. Additionally, communicate openly with employees about the rationale behind adopting the solution, its potential impact on their roles, and how it aligns with the organization's broader goals. By involving employees early in the process and addressing any concerns or resistance to change, you can facilitate a smoother transition and foster greater acceptance of the new technology.
3. **Continuous Evaluation and Optimization:** The implementation of an automated support ticket categorization solution is not a one-time endeavor; it requires ongoing evaluation and optimization to ensure its effectiveness and relevance over time. Establish mechanisms for continuous monitoring of key performance indicators (KPIs), such as categorization accuracy, ticket resolution times, and customer satisfaction scores. Regularly review the system's outputs, analyze any discrepancies or patterns in ticket categorization, and gather feedback from users to identify areas for improvement. Consider implementing regular calibration sessions where support teams can provide input on the categorization results and collaborate on refining the system's algorithms and rules. By embracing a culture of continuous evaluation and optimization, you can drive continuous improvement in support ticket management processes and deliver enhanced value to both customers and the organization.

By following these recommendations, businesses can successfully adopt and leverage an automated support ticket categorization solution to streamline their support operations, improve efficiency, and deliver superior customer experiences.

Notebook Overall Quality

Structure and flow - Well-commented code

- **Business Context:** I have provided a comprehensive overview of why automated support ticket categorization is crucial in today's business landscape. It emphasizes the importance of customer feedback and outlines the strategic imperative for Product Managers or Analysts to stay attuned to customer needs.
- **Objective:** I have clearly stated the objective of the notebook, which is to develop a Generative AI application using a Large Language Model to automate the classification and processing of support tickets.
- **Data Overview:** This section involves loading the dataset and providing an overview of its structure and content.
- **Model Building:** I have loaded a pre-trained model from Hugging Face and defined a function to generate responses from the model.
- **Tasks 1-6:** Each task involves progressively more complex ticket categorization and processing, starting from basic categorization to generating a draft response based on the ticket content.
- **Model Output Analysis:** After processing the tickets, i have performed univariate analysis on the structured output DataFrame to derive insights from the data.
- **Actionable Insights and Recommendations:** Finally, I have shared observations, insights, and recommendations based on the results obtained from the automated ticket processing.

Overall, this notebook appears to be well-structured, covering all essential aspects of automating support ticket categorization and providing actionable insights for businesses looking to adopt such a solution. It's designed to guide users through each step of the process with clear instructions and code examples.