```
!CMAKE_ARGS="-DLLAMA_CUBLAS=on" FORCE_CMAKE=1 pip install llama-cpp-python==0.2.45 --force-reinstall --no-cache-
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 36.7/36.7 MB 227.0 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
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.1/62.1 kB 246.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.5/45.5 kB 217.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.9/134.9 kB 244.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.6/16.6 MB 231.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.6/44.6 kB 150.6 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 that are installed. This b
opencv-contrib-python 4.12.0.88 requires numpy<2.3.0,>=2; python_version >= "3.9", but you have numpy 2.3.4 which
opencv-python 4.12.0.88 requires numpy<2.3.0,>=2; python_version >= "3.9", but you have numpy 2.3.4 which is inco
tensorflow 2.19.0 requires numpy<2.2.0,>=1.26.0, but you have numpy 2.3.4 which is incompatible.
opencv-python-headless 4.12.0.88 requires numpy<2.3.0,>=2; python_version >= "3.9", but you have numpy 2.3.4 whic
numba 0.60.0 requires numpy<2.1,>=1.22, but you have numpy 2.3.4 which is incompatible.
cupy-cuda12x 13.3.0 requires numpy<2.3,>=1.22, but you have numpy 2.3.4 which is incompatible.
```

```
!pip install huggingface_hub
```

```
Requirement already satisfied: huggingface_hub in /usr/local/lib/python3.12/dist-packages (0.36.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (3.20.0
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub)
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (6.0
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (2.32.4
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (4.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from huggin
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface_
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->huggingfac
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->hugg
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->hugg
```

```
import pandas as pd
from huggingface_hub import hf_hub_download
from llama_cpp import Llama
import json
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force
```

```
data = pd.read_csv("/content/drive/MyDrive/restaurant_reviews.csv")
```

```
data.head()
```

	restaurant_ID	rating_review	review_full
0	FLV202	5	Totally in love with the Auro of the place, re...
1	SAV303	5	Kailash colony is brimming with small cafes no...
2	YUM789	5	Excellent taste and awesome decorum. Must visi...
3	TST101	5	I have visited at jw lough/restourant. There w...
4	EAT456	5	Had a great experience in the restaurant food ...

Next steps: Generate code with data New interactive sheet

```
data['review_full'][3]
```

```
'I have visited at jw lough/restourant. There were a first class service at lough, specially Ms.laxmi  who were
superbed for handling the client need, me and my family lots enjoyed her specialty in the manner, and Laxmi is a
very very good in the client service, I hope when I will come against I would definitely serve from Ms. Laxmi an
d she is wonderful girl in that service. See you again Ms. Laxmi for the your best service which I have received
```

```
data.shape
```

```
(20, 3)
```

```
data.isnull().sum()
```

	0
restaurant_ID	0
rating_review	0
review_full	0

dtype: int64

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

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

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: 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://huggingface.co/settings/t
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(
```

llama-2-13b-chat.Q5_K_M.gguf: 100% 9.23G/9.23G [01:14<00:00, 231MB/s]

```
Llama_llm = Llama(
    model_path=model_path,
    n_threads=2,
    n_batch=512,
    n_gpu_layers=43,
    n_ctx=4096,
)
```

```
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: n_ff             = 13824
llm_load_print_meta: n_expert         = 0
llm_load_print_meta: n_expert_used    = 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 input buffer size   =    19.04 MiB
llama_new_context_with_model:      CUDA0 compute buffer size =   368.02 MiB
llama_new_context_with_model:  CUDA_Host compute buffer size =    10.00 MiB
llama_new_context_with_model: graph splits (measure): 3
AVX = 1 | AVX_VNNI = 0 | AVX2 = 1 | AVX512 = 1 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | FMA = 1 | NEON = 0 | ARM_FM
Model metadata: {'tokenizer.ggml.unknown_token_id': '0', 'tokenizer.ggml.eos_token_id': '2', 'general.architectu
```

```
model_name_or_path = "TheBloke/Mistral-7B-Instruct-v0.2-GGUF"
model_basename = "mistral-7b-instruct-v0.2.Q6_K.gguf"
```

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

```
mistral-7b-instruct-v0.2.Q6_K.gguf: 100%                                5.94G/5.94G [01:26<00:00, 250MB/s]
```

```
mistral_llm = Llama(
    model_path=model_path,n_ctx=1024)
```

```
llama_model_loader: loaded meta data with 24 key-value pairs and 291 tensors from /root/.cache/huggingface/hub/m
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              = mistralai_mistral-7b
llama_model_loader: - kv   2:                       llama.context_length u32              = 32768
llama_model_loader: - kv   3:                     llama.embedding_length u32              = 4096
llama_model_loader: - kv   4:                          llama.block_count u32              = 32
llama_model_loader: - kv   5:                  llama.feed_forward_length u32              = 14336
llama_model_loader: - kv   6:                 llama.rope.dimension_count u32              = 128
llama_model_loader: - kv   7:                 llama.attention.head_count u32              = 32
llama_model_loader: - kv   8:              llama.attention.head_count_kv u32              = 8
llama_model_loader: - kv   9:     llama.attention.layer_norm_rms_epsilon f32              = 0.000010
llama_model_loader: - kv  10:                       llama.rope.freq_base f32              = 1000000.000000
llama_model_loader: - kv  11:                          general.file_type u32              = 18
llama_model_loader: - kv  12:                       tokenizer.ggml.model str              = llama
llama_model_loader: - kv  13:                      tokenizer.ggml.tokens arr[str,32000]   = ["<unk>", "<s>", "</
llama_model_loader: - kv  14:                      tokenizer.ggml.scores arr[f32,32000]   = [0.000000, 0.000000,
llama_model_loader: - kv  15:                  tokenizer.ggml.token_type arr[i32,32000]   = [2, 3, 3, 6, 6, 6, 6
llama_model_loader: - kv  16:                tokenizer.ggml.bos_token_id u32              = 1
llama_model_loader: - kv  17:                tokenizer.ggml.eos_token_id u32              = 2
llama_model_loader: - kv  18:            tokenizer.ggml.unknown_token_id u32              = 0
llama_model_loader: - kv  19:            tokenizer.ggml.padding_token_id u32              = 0
llama_model_loader: - kv  20:               tokenizer.ggml.add_bos_token bool             = true
llama_model_loader: - kv  21:               tokenizer.ggml.add_eos_token bool             = false
llama_model_loader: - kv  22:                    tokenizer.chat_template str              = {{ bos_token }}{% fo
llama_model_loader: - kv  23:               general.quantization_version u32              = 2
llama_model_loader: - type  f32:   65 tensors
llama_model_loader: - type q6_K:  226 tensors
llm_load_vocab: special tokens definition check successful ( 259/32000 ).
llm_load_print_meta: format           = GGUF V3 (latest)
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      = 32768
llm_load_print_meta: n_embd           = 4096
llm_load_print_meta: n_head           = 32
llm_load_print_meta: n_head_kv        = 8
llm_load_print_meta: n_layer          = 32
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            = 4
llm_load_print_meta: n_embd_k_gqa     = 1024
llm_load_print_meta: n_embd_v_gqa     = 1024
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: n_ff             = 14336
llm_load_print_meta: n_expert         = 0
llm_load_print_meta: n_expert_used    = 0
llm_load_print_meta: rope scaling     = linear
llm_load_print_meta: freq_base_train  = 1000000.0
llm_load_print_meta: freq_scale_train = 1
llm_load_print_meta: n_yarn_orig_ctx  = 32768
llm_load_print_meta: rope_finetuned   = unknown
```

```
def generate_llama_response(instruction, review):
    system_message = """
        [INST]<<SYS>>
```

```
            {}
            <</SYS>>[/INST]
        """.format(instruction)
        prompt = f"{review}\n{system_message}"
        response = Llama_llm(
            prompt=prompt,
            max_tokens=1024,
            temperature=0,
            top_p=0.95,
            repeat_penalty=1.2,
            top_k=50,
            stop=['INST'],
            echo=False,
            seed=42,
        )
        response_text = response["choices"][0]["text"]
        return response_text
```

```
    def extract_json_data(json_str):
        try:
            json_start = json_str.find('{')
            json_end = json_str.rfind('}')

            if json_start != -1 and json_end != -1:
                extracted_sentiment = json_str[json_start:json_end + 1]
                data_dict = json.loads(extracted_sentiment)
                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 {}
```

```
    data_1 = data.copy()
```

```
    instruction_1 = """
        You are an AI analyzing restaurant reviews. Classify the sentiment of the provided review into the following
        - Positive
        - Negative
        - Neutral
    """
    data_1['model_response'] = data_1['review_full'].apply(lambda x: generate_llama_response(instruction_1, x))
```

```
llama_print_timings: prompt eval time =     905.47 ms /   415 tokens (    2.18 ms per token,   458.32 tokens pe
llama_print_timings:        eval time =    8547.59 ms /   133 runs   (   64.27 ms per token,    15.56 tokens pe
llama_print_timings:       total time =   10054.68 ms /   548 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      60.78 ms /   113 runs   (    0.54 ms per token,  1859.10 tokens pe
llama_print_timings: prompt eval time =    1394.80 ms /   545 tokens (    2.56 ms per token,   390.74 tokens pe
llama_print_timings:        eval time =    7464.78 ms /   112 runs   (   66.65 ms per token,    15.00 tokens pe
llama_print_timings:       total time =    9262.05 ms /   657 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     108.27 ms /   171 runs   (    0.63 ms per token,  1579.33 tokens pe
llama_print_timings: prompt eval time =     819.95 ms /   368 tokens (    2.23 ms per token,   448.81 tokens pe
llama_print_timings:        eval time =   11092.34 ms /   170 runs   (   65.25 ms per token,    15.33 tokens pe
llama_print_timings:       total time =   12612.03 ms /   538 tokens
```

```
data_1['model_response'].head()
```

	model_response
0	Sure! Here's the sentiment analysis of the re...
1	Sure! Here's the sentiment analysis of the pr...
2	Sure, I can help you with that! Based on the ...
3	Sure! Here's the classification of the review...
4	Sure! Here's the sentiment analysis of the re...

dtype: object

```
data.shape
```

```
(20, 3)
```

```
i = 2
print(data_1.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had given us a great service. One of the best exp
```

```
print(data_1.loc[i, 'model_response'])
```

```
 Sure, I can help you with that! Based on the review you provided, I would classify it as:

Positive

Reason: The reviewer enjoyed their experience at the restaurant, mentioning that it has "excellent taste" and "aw
```

```
def extract_sentiment(model_response):
    if 'positive' in model_response.lower():
        return 'Positive'
    elif 'negative' in model_response.lower():
        return 'Negative'
    elif 'neutral' in model_response.lower():
        return 'Neutral'
```

```
data_1['sentiment'] = data_1['model_response'].apply(extract_sentiment)
data_1['sentiment'].head()
```

	sentiment
0	Positive
1	Positive
2	Positive
3	Positive
4	Positive

dtype: object

```
data_1['sentiment'].value_counts()
```

sentiment	count
Positive	16
Negative	3
Neutral	1

dtype: int64

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

	restaurant_ID	rating_review	review_full	sentiment
0	FLV202	5	Totally in love with the Auro of the place, re...	Positive
1	SAV303	5	Kailash colony is brimming with small cafes no...	Positive
2	YUM789	5	Excellent taste and awesome decorum. Must visi...	Positive
3	TST101	5	I have visited at jw lough/restourant. There w...	Positive
4	EAT456	5	Had a great experience in the restaurant food ...	Positive

Next steps: Generate code with final_data_1 New interactive sheet

```
data_1 = data.copy()
```

```
def response_1(prompt,review):
    model_output = mistral_llm(
      f"""
      Q: {prompt}
      Review: {review}
      A:
      """,
      max_tokens=32,
      stop=["Q:", "\n"],
      temperature=0.01,
      echo=False,
    )

    temp_output = model_output["choices"][0]["text"]

    return temp_output
```

```
instruction_1 = """
    You are an AI analyzing restaurant reviews. Classify the sentiment of the provided review into the following
    - Positive
    - Negative
    - Neutral
"""
data_1['model_response'] = data_1['review_full'].apply(lambda x: response_1(instruction_1, x))
```

```
llama_print_timings:        load time =    3231.24 ms
llama_print_timings:      sample time =      17.90 ms /    32 runs   (    0.56 ms per token,  1787.61 tokens pe
llama_print_timings: prompt eval time =    3230.77 ms /   218 tokens (   14.82 ms per token,    67.48 tokens pe
llama_print_timings:        eval time =   25904.22 ms /    31 runs   (  835.62 ms per token,     1.20 tokens pe
llama_print_timings:       total time =   29291.17 ms /   249 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3231.24 ms
llama_print_timings:      sample time =      18.75 ms /    32 runs   (    0.59 ms per token,  1706.39 tokens pe
llama_print_timings: prompt eval time =    2582.15 ms /   235 tokens (   10.99 ms per token,    91.01 tokens pe
llama_print_timings:        eval time =   25810.23 ms /    31 runs   (  832.59 ms per token,     1.20 tokens pe
llama_print_timings:       total time =   28551.42 ms /   266 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3231.24 ms
llama_print_timings:      sample time =      18.36 ms /    32 runs   (    0.57 ms per token,  1742.82 tokens pe
llama_print_timings: prompt eval time =    1612.26 ms /    34 tokens (   47.42 ms per token,    21.09 tokens pe
llama_print_timings:        eval time =   27038.94 ms /    31 runs   (  872.22 ms per token,     1.15 tokens pe
llama_print_timings:       total time =   28823.19 ms /    65 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3231.24 ms
llama_print_timings:      sample time =      18.37 ms /    32 runs   (    0.57 ms per token,  1741.69 tokens pe
llama_print_timings: prompt eval time =    2127.28 ms /   143 tokens (   14.88 ms per token,    67.22 tokens pe
llama_print_timings:        eval time =   25231.24 ms /    31 runs   (  813.91 ms per token,     1.23 tokens pe
llama_print_timings:       total time =   27503.01 ms /   174 tokens
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3231.24 ms
llama_print_timings:      sample time =      17.77 ms /    32 runs   (    0.56 ms per token,  1800.69 tokens pe
llama_print_timings: prompt eval time =    2321.19 ms /   169 tokens (   13.73 ms per token,    72.81 tokens pe
llama_print_timings:        eval time =   25000.84 ms /    31 runs   (  806.48 ms per token,     1.24 tokens pe
llama_print_timings:       total time =   27459.52 ms /   200 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3231.24 ms
llama_print_timings:      sample time =      17.98 ms /    32 runs   (    0.56 ms per token,  1779.26 tokens pe
llama_print_timings: prompt eval time =    3378.89 ms /   217 tokens (   15.57 ms per token,    64.22 tokens pe
llama_print_timings:        eval time =   25035.84 ms /    31 runs   (  807.61 ms per token,     1.24 tokens pe
llama_print_timings:       total time =   28558.78 ms /   248 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3231.24 ms
llama_print_timings:      sample time =      17.79 ms /    32 runs   (    0.56 ms per token,  1798.36 tokens pe
llama_print_timings: prompt eval time =    2745.94 ms /   215 tokens (   12.77 ms per token,    78.30 tokens pe
llama_print_timings:        eval time =   25222.33 ms /    31 runs   (  813.62 ms per token,     1.23 tokens pe
llama_print_timings:       total time =   28109.09 ms /   246 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3231.24 ms
llama_print_timings:      sample time =      18.52 ms /    32 runs   (    0.58 ms per token,  1727.96 tokens pe
llama_print_timings: prompt eval time =    1743.23 ms /    69 tokens (   25.26 ms per token,    39.58 tokens pe
llama_print_timings:        eval time =   24848.50 ms /    31 runs   (  801.56 ms per token,     1.25 tokens pe
llama_print_timings:       total time =   26737.82 ms /   100 tokens
Llama.generate: prefix-match hit
```

```
data_1['model_response'].head()
```

	model_response
0	This review expresses a very positive sentime...
1	Based on the given review, the sentiment can ...
2	The sentiment expressed in this review is pos...
3	Based on the given review, it can be classifi...
4	Based on the provided review, the sentiment c...

dtype: object

```
i = 2
print(data_1.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had given us a great service. One of the best exp
```

```
print(data_1.loc[i, 'model_response'])
```

```
 The sentiment expressed in this review is positive. The use of words like "excellent taste," "awesome decorum,"
```

```
def extract_sentiment(model_response):
    if 'positive' in model_response.lower():
        return 'Positive'
    elif 'negative' in model_response.lower():
        return 'Negative'
    elif 'neutral' in model_response.lower():
        return 'Neutral'
```

```
data_1['sentiment'] = data_1['model_response'].apply(extract_sentiment)
data_1['sentiment'].head()
```

	sentiment
0	Positive
1	Positive
2	Positive
3	Positive
4	Positive

dtype: object

```
data_1['sentiment'].value_counts()
```

sentiment	count
Positive	10
Negative	7
Neutral	3

dtype: int64

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

	restaurant_ID	rating_review	review_full	sentiment
0	FLV202	5	Totally in love with the Auro of the place, re...	Positive
1	SAV303	5	Kailash colony is brimming with small cafes no...	Positive
2	YUM789	5	Excellent taste and awesome decorum. Must visi...	Positive
3	TST101	5	I have visited at jw lough/restourant. There w...	Positive
4	EAT456	5	Had a great experience in the restaurant food ...	Positive

Next steps: Generate code with final_data_1 New interactive sheet

```
data_2 = data.copy()
```

```
instruction_2 = """
    You are an AI analyzing restaurant reviews. Classify the sentiment of the provided review into the following
    - Positive
    - Negative
    - Neutral

    Format the output as a JSON object with a single key-value pair as shown below:
    {"sentiment": "your_sentiment_prediction"}
"""
data_2['model_response'] = data_2['review_full'].apply(lambda x: generate_llama_response(instruction_2, x))
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      95.19 ms /   161 runs   (    0.59 ms per token,  1691.34 tokens pe
llama_print_timings: prompt eval time =     787.90 ms /   272 tokens (    2.90 ms per token,   345.22 tokens pe
llama_print_timings:        eval time =   11384.98 ms /   160 runs   (   71.16 ms per token,    14.05 tokens pe
llama_print_timings:       total time =   12826.72 ms /   432 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     131.50 ms /   225 runs   (    0.58 ms per token,  1711.03 tokens pe
llama_print_timings: prompt eval time =     875.32 ms /   343 tokens (    2.55 ms per token,   391.86 tokens pe
llama_print_timings:        eval time =   16024.26 ms /   224 runs   (   71.54 ms per token,    13.98 tokens pe
llama_print_timings:       total time =   17818.83 ms /   567 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      30.64 ms /    48 runs   (    0.64 ms per token,  1566.63 tokens pe
llama_print_timings: prompt eval time =     544.05 ms /   133 tokens (    4.09 ms per token,   244.46 tokens pe
llama_print_timings:        eval time =    3022.04 ms /    47 runs   (   64.30 ms per token,    15.55 tokens pe
llama_print_timings:       total time =    3774.89 ms /   180 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      58.82 ms /   109 runs   (    0.54 ms per token,  1852.99 tokens pe
llama_print_timings: prompt eval time =     615.32 ms /   243 tokens (    2.53 ms per token,   394.92 tokens pe
llama_print_timings:        eval time =    7046.46 ms /   108 runs   (   65.24 ms per token,    15.33 tokens pe
llama_print_timings:       total time =    8032.86 ms /   351 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     141.90 ms /   238 runs   (    0.60 ms per token,  1677.30 tokens pe
llama_print_timings: prompt eval time =     783.57 ms /   280 tokens (    2.80 ms per token,   357.34 tokens pe
llama_print_timings:        eval time =   14556.04 ms /   237 runs   (   61.42 ms per token,    16.28 tokens pe
llama_print_timings:       total time =   16367.18 ms /   517 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      74.74 ms /   130 runs   (    0.57 ms per token,  1739.27 tokens pe
llama_print_timings: prompt eval time =     783.35 ms /   323 tokens (    2.43 ms per token,   412.33 tokens pe
llama_print_timings:        eval time =    8073.12 ms /   129 runs   (   62.58 ms per token,    15.98 tokens pe
llama_print_timings:       total time =    9340.02 ms /   452 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     171.34 ms /   259 runs   (    0.66 ms per token,  1511.59 tokens pe
llama_print_timings: prompt eval time =     796.08 ms /   329 tokens (    2.42 ms per token,   413.28 tokens pe
llama_print_timings:        eval time =   15933.89 ms /   258 runs   (   61.76 ms per token,    16.19 tokens pe
llama_print_timings:       total time =   18011.46 ms /   587 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      50.32 ms /    90 runs   (    0.56 ms per token,  1788.48 tokens pe
llama_print_timings: prompt eval time =     580.78 ms /   168 tokens (    3.46 ms per token,   289.27 tokens pe
llama_print_timings:        eval time =    5714.66 ms /    89 runs   (   64.21 ms per token,    15.57 tokens pe
llama_print_timings:       total time =    6662.36 ms /   257 tokens
Llama.generate: prefix-match hit
```

```
data_2['model_response'].head()
```

	model_response
0	Sure! Here's the sentiment analysis of the re...
1	Sure! Here's my analysis of the review you pr...
2	Sure, I can help you with that! Based on the ...
3	Sure! Here's the sentiment analysis of the re...
4	Sure! Here's the sentiment analysis of the re...

dtype: object

```
i = 2
print(data_2.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had given us a great service. One of the best exp
```

```
print(data_2.loc[i, 'model_response'])
```

```
 Sure, I can help you with that! Based on the review you provided, I would classify the sentiment as Positive. He

{"sentiment": "Positive"}
```

```
data_2['model_response_parsed'] = data_2['model_response'].apply(extract_json_data)
data_2['model_response_parsed'].head()
```

	model_response_parsed
0	{'sentiment': 'Positive'}
1	{'sentiment': 'Positive'}
2	{'sentiment': 'Positive'}
3	{'sentiment': 'Positive'}
4	{'sentiment': 'Positive'}

dtype: object

```
model_response_parsed_df_2 = pd.json_normalize(data_2['model_response_parsed'])
model_response_parsed_df_2.head()
```

	sentiment
0	Positive
1	Positive
2	Positive
3	Positive
4	Positive

Next steps: Generate code with `model_response_parsed_df_2` New interactive sheet

```
data_with_parsed_model_output_2 = pd.concat([data_2, model_response_parsed_df_2], axis=1)
data_with_parsed_model_output_2.head()
```

	restaurant_ID	rating_review	review_full	model_response	model_response_parsed	sentiment
0	FLV202	5	Totally in love with the Auro of the place, re...	Sure! Here's the sentiment analysis of the re...	{'sentiment': 'Positive'}	Positive
1	SAV303	5	Kailash colony is brimming with small cafes no...	Sure! Here's my analysis of the review you pr...	{'sentiment': 'Positive'}	Positive
2	YUM789	5	Excellent taste and awesome decorum. Must visi...	Sure, I can help you with that! Based on the ...	{'sentiment': 'Positive'}	Positive

Next steps: Generate code with data_with_parsed_model_output_2 New interactive sheet

```
final_data_2 = data_with_parsed_model_output_2.drop(['model_response','model_response_parsed'], axis=1)
final_data_2.head()
```

	restaurant_ID	rating_review	review_full	sentiment
0	FLV202	5	Totally in love with the Auro of the place, re...	Positive
1	SAV303	5	Kailash colony is brimming with small cafes no...	Positive
2	YUM789	5	Excellent taste and awesome decorum. Must visi...	Positive
3	TST101	5	I have visited at jw lough/restourant. There w...	Positive
4	EAT456	5	Had a great experience in the restaurant food ...	Positive

Next steps: Generate code with final_data_2 New interactive sheet

```
final_data_2['sentiment'].value_counts()
```

sentiment	count
Neutral	7
Negative	7
Positive	6

dtype: int64

```
data_3 = data.copy()
```

```
instruction_3 = """
    You are an AI analyzing restaurant reviews. Classify the overall sentiment of the provided review into the f
    - "Positive"
    - "Negative"
    - "Neutral"

    Once that is done, check for a mention of the following aspects in the review and classify the sentiment of
    1. "Food Quality"
    2. "Service"
    3. "Ambience"

    Output the overall sentiment and sentiment for each category in a JSON format with the following keys:
    {
        "Overall": "your_sentiment_prediction",
        "Food Quality": "your_sentiment_prediction",
        "Service": "your_sentiment_prediction",
        "Ambience": "your_sentiment_prediction"
    }

    In case one of the three aspects is not mentioned in the review, set "Not Applicable" (including quotes) for
    Only return the JSON, do not return any other information.
"""
data_3['model_response'] = data_3['review_full'].apply(lambda x: generate_llama_response(instruction_3, x))
```

```
llama_print_timings:      sample time =      28.96 ms /    52 runs   (    0.56 ms per token,  1795.70 tokens pe
llama_print_timings: prompt eval time =    1592.46 ms /   650 tokens (    2.45 ms per token,   408.17 tokens pe
llama_print_timings:        eval time =    3485.70 ms /    51 runs   (   68.35 ms per token,    14.63 tokens pe
llama_print_timings:       total time =    5258.84 ms /   701 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     726.39 ms /  1024 runs   (    0.71 ms per token,  1409.71 tokens pe
llama_print_timings: prompt eval time =    1864.57 ms /   821 tokens (    2.27 ms per token,   440.32 tokens pe
llama_print_timings:        eval time =   70178.34 ms /  1023 runs   (   68.60 ms per token,    14.58 tokens pe
llama_print_timings:       total time =   77923.04 ms /  1844 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      55.48 ms /    52 runs   (    1.07 ms per token,   937.29 tokens pe
llama_print_timings: prompt eval time =    1399.74 ms /   562 tokens (    2.49 ms per token,   401.50 tokens pe
llama_print_timings:        eval time =    3227.35 ms /    51 runs   (   63.28 ms per token,    15.80 tokens pe
llama_print_timings:       total time =    4977.40 ms /   613 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      28.38 ms /    53 runs   (    0.54 ms per token,  1867.31 tokens pe
llama_print_timings: prompt eval time =    1403.31 ms /   566 tokens (    2.48 ms per token,   403.33 tokens pe
llama_print_timings:        eval time =    3496.35 ms /    52 runs   (   67.24 ms per token,    14.87 tokens pe
llama_print_timings:       total time =    5083.36 ms /   618 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     192.55 ms /   294 runs   (    0.65 ms per token,  1526.88 tokens pe
llama_print_timings: prompt eval time =    1487.58 ms /   630 tokens (    2.36 ms per token,   423.51 tokens pe
llama_print_timings:        eval time =   19143.39 ms /   293 runs   (   65.34 ms per token,    15.31 tokens pe
llama_print_timings:       total time =   22090.57 ms /   923 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      27.51 ms /    51 runs   (    0.54 ms per token,  1854.07 tokens pe
llama_print_timings: prompt eval time =    1671.54 ms /   760 tokens (    2.20 ms per token,   454.67 tokens pe
llama_print_timings:        eval time =    3383.35 ms /    50 runs   (   67.67 ms per token,    14.78 tokens pe
llama_print_timings:       total time =    5238.71 ms /   810 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      36.18 ms /    51 runs   (    0.71 ms per token,  1409.62 tokens pe
llama_print_timings: prompt eval time =    1435.87 ms /   583 tokens (    2.46 ms per token,   406.03 tokens pe
llama_print_timings:        eval time =    3252.44 ms /    50 runs   (   65.05 ms per token,    15.37 tokens pe
llama_print_timings:       total time =    4933.22 ms /   633 tokens
```

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

	model_response
0	\n {\n "Overall": "Positive",\n ...
1	\n {\n "Overall": "Positive",\n ...
2	{\n "Overall": "Positive",\n ...
3	{\n "Overall": "Positive",\n "F...
4	\n {\n "Overall": "Positive",\n ...

dtype: object

```
i = 2
print(data_3.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had given us a great service. One of the best exp
```

```
print(data_3.loc[i, 'model_response'])
```

```
 {
           "Overall": "Positive",
           "Food Quality": "Positive",
           "Service": "Positive",
           "Ambience": "Positive"
         }
```

```
data_3['model_response_parsed'] = data_3['model_response'].apply(extract_json_data)
data_3['model_response_parsed'].head()
```

	model_response_parsed
0	{'Overall': 'Positive', 'Food Quality': 'Posit...
1	{'Overall': 'Positive', 'Food Quality': 'Posit...
2	{'Overall': 'Positive', 'Food Quality': 'Posit...
3	{'Overall': 'Positive', 'Food Quality': 'Posit...
4	{'Overall': 'Positive', 'Food Quality': 'Posit...

dtype: object

```
model_response_parsed_df_3 = pd.json_normalize(data_3['model_response_parsed'])
model_response_parsed_df_3.head()
```

	Overall	Food Quality	Service	Ambience
0	Positive	Positive	Positive	Positive
1	Positive	Positive	Neutral	Positive
2	Positive	Positive	Positive	Positive
3	Positive	Positive	Positive	Positive
4	Positive	Positive	Positive	Positive

Next steps: Generate code with model_response_parsed_df_3 New interactive sheet

```
data_with_parsed_model_output_3 = pd.concat([data_3, model_response_parsed_df_3], axis=1)
data_with_parsed_model_output_3.head()
```

	restaurant_ID	rating_review	review_full	model_response	model_response_parsed	Overall	Food Quality	Service	Am
0	FLV202	5	Totally in love with the Auro of the place, re...	\n {\n "Overall": "Positive",\n ...	{'Overall': 'Positive', 'Food Quality': 'Posit...	Positive	Positive	Positive	
1	SAV303	5	Kailash colony is brimming with small cafes no...	\n {\n "Overall": "Positive",\n ...	{'Overall': 'Positive', 'Food Quality': 'Posit...	Positive	Positive	Neutral	
2	YUM789	5	Excellent taste and awesome decorum. Must visi...	{\n "Overall": "Positive",\n ...	{'Overall': 'Positive', 'Food Quality': 'Posit...	Positive	Positive	Positive	
			I have visited at						

Next steps: Generate code with data_with_parsed_model_output_3 New interactive sheet

```
final_data_3 = data_with_parsed_model_output_3.drop(['model_response','model_response_parsed'], axis=1)
final_data_3.head()
```

	restaurant_ID	rating_review	review_full	Overall	Food Quality	Service	Ambience
0	FLV202	5	Totally in love with the Auro of the place, re...	Positive	Positive	Positive	Positive
1	SAV303	5	Kailash colony is brimming with small cafes no...	Positive	Positive	Neutral	Positive
2	YUM789	5	Excellent taste and awesome decorum. Must visi...	Positive	Positive	Positive	Positive
			I have visited at iw lough/restourant. There				

Next steps: Generate code with final_data_3 New interactive sheet

```
final_data_3['Overall'].value_counts()
```

Overall	count
Neutral	7
Negative	7
Positive	6

dtype: int64

```
final_data_3['Food Quality'].value_counts()
```

Food Quality	count
Positive	7
Neutral	7
Negative	3
Mixed	2
Not Applicable	1

dtype: int64

```
final_data_3['Service'].value_counts()
```

Service	count
Negative	9
Positive	8
Neutral	1
Slow	1
Inconsistent	1

dtype: int64

```
final_data_3['Ambience'].value_counts()
```

Ambience	count
Positive	9
Not Applicable	6
Neutral	4
Negative	1

dtype: int64

```
data_3 = data.copy()
```

```
def response_2(prompt,review,sentiment):
    model_output = llm(
      f"""
      Q: {prompt}
      review: {review}
      sentiment: {sentiment}
      A:
      """,
      max_tokens=64,
      stop=["Q:", "\n"],
      temperature=0.01,
      echo=False,
    )

    temp_output = model_output["choices"][0]["text"]
    final_output = temp_output[temp_output.index('{'):]

    return final_output
```

```
i = 2
print(data_3.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had given us a great service. One of the best exp
```

```
data_4 = data.copy()
```

```
instruction_4 = """
    You are an AI tasked with analyzing restaurant reviews. Your goal is to classify the overall sentiment of th
        - Positive
        - Negative
        - Neutral

    Subsequently, assess the sentiment of specific aspects mentioned in the review, namely:
        1. Food quality
        2. Service
        3. Ambience

    Further, identify liked and/or disliked features associated with each aspect in the review.

    Return the output in the specified JSON format, ensuring consistency and handling missing values appropriate

    {
        "Overall": "your_sentiment_prediction",
        "Food Quality": "your_sentiment_prediction",
        "Service": "your_sentiment_prediction",
        "Ambience": "your_sentiment_prediction",
        "Food Quality Features": ["liked/disliked features"],
        "Service Features": ["liked/disliked features"],
        "Ambience Features": ["liked/disliked features"]
    }

    The sentiment prediction for Overall, Food Quality, Service, and Ambience should be one of "Positive", "Nega
    In case one of the three aspects is not mentioned in the review, set "Not Applicable" (including quotes) in
    In case there are no liked/disliked features for a particular aspect, assign an empty list in the correspond
    Only return the JSON, do NOT return any other text or information.
"""
```

```
data_4['model_response'] = data_4['review_full'].apply(lambda x: generate_llama_response(instruction_4, x).repla
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     172.41 ms /   294 runs   (    0.59 ms per token,  1705.27 tokens pe
llama_print_timings: prompt eval time =    1519.81 ms /   574 tokens (    2.65 ms per token,   377.68 tokens pe
llama_print_timings:        eval time =   24128.55 ms /   293 runs   (   82.35 ms per token,    12.14 tokens pe
llama_print_timings:       total time =   26902.72 ms /   867 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     139.53 ms /   246 runs   (    0.57 ms per token,  1763.01 tokens pe
llama_print_timings: prompt eval time =    1637.54 ms /   645 tokens (    2.54 ms per token,   393.88 tokens pe
llama_print_timings:        eval time =   16371.85 ms /   245 runs   (   66.82 ms per token,    14.96 tokens pe
llama_print_timings:       total time =   19035.10 ms /   890 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     178.54 ms /   281 runs   (    0.64 ms per token,  1573.90 tokens pe
llama_print_timings: prompt eval time =     897.01 ms /   435 tokens (    2.06 ms per token,   484.94 tokens pe
llama_print_timings:        eval time =   17326.00 ms /   280 runs   (   61.88 ms per token,    16.16 tokens pe
llama_print_timings:       total time =   19452.90 ms /   715 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     202.09 ms /   345 runs   (    0.59 ms per token,  1707.14 tokens pe
llama_print_timings: prompt eval time =    1356.65 ms /   545 tokens (    2.49 ms per token,   401.72 tokens pe
llama_print_timings:        eval time =   22248.71 ms /   344 runs   (   64.68 ms per token,    15.46 tokens pe
llama_print_timings:       total time =   25095.42 ms /   889 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     129.55 ms /   227 runs   (    0.57 ms per token,  1752.22 tokens pe
llama_print_timings: prompt eval time =    1440.61 ms /   582 tokens (    2.48 ms per token,   404.00 tokens pe
llama_print_timings:        eval time =   15378.37 ms /   226 runs   (   68.05 ms per token,    14.70 tokens pe
llama_print_timings:       total time =   17734.40 ms /   808 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     171.16 ms /   274 runs   (    0.62 ms per token,  1600.79 tokens pe
llama_print_timings: prompt eval time =    1519.91 ms /   625 tokens (    2.43 ms per token,   411.21 tokens pe
llama_print_timings:        eval time =   18521.71 ms /   273 runs   (   67.85 ms per token,    14.74 tokens pe
llama_print_timings:       total time =   21235.06 ms /   898 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     195.20 ms /   335 runs   (    0.58 ms per token,  1716.21 tokens pe
llama_print_timings: prompt eval time =    1504.16 ms /   631 tokens (    2.38 ms per token,   419.50 tokens pe
llama_print_timings:        eval time =   22364.34 ms /   334 runs   (   66.96 ms per token,    14.93 tokens pe
llama_print_timings:       total time =   25281.73 ms /   965 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     158.88 ms /   274 runs   (    0.58 ms per token,  1724.62 tokens pe
llama_print_timings: prompt eval time =     952.89 ms /   470 tokens (    2.03 ms per token,   493.23 tokens pe
llama_print_timings:        eval time =   17981.93 ms /   273 runs   (   65.87 ms per token,    15.18 tokens pe
llama_print_timings:       total time =   20024.84 ms /   743 tokens
Llama.generate: prefix-match hit
```

```
i = 2
print(data_4.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had given us a great service. One of the best exp
```

```
print(data_4.loc[i, 'model_response'])
```

```
 Sure, I can assist you in that! Here's my analysis of your provided review: {"Overall": "Positive", "Food Qualit
```

```
data_4['model_response_parsed'] = data_4['model_response'].apply(extract_json_data)
data_4['model_response_parsed'].head()
```

	model_response_parsed
0	{'Overall': 'Positive', 'Food Quality': 'Posit...
1	{'Overall': 'Positive', 'Food Quality': 'Posit...
2	{'Overall': 'Positive', 'Food Quality': 'Posit...
3	{'Overall': 'Positive', 'Food Quality': 'Posit...
4	{'Overall': 'Positive', 'Food Quality': 'Posit...

dtype: object

```
data_4[data_4.model_response_parsed == {}]
```

restaurant_ID	rating_review	review_full	model_response	model_response_parsed

```
print(data_4.loc[3, 'model_response'])
```

```
 Sure! Here's the JSON output based on your review:{    "Overall": "Positive",    "Food Quality": "Positive",
```

```
print(data_4.loc[6, 'model_response'])
```

```
    Sure! Here's my analysis of the review you provided:    {        "Overall": "Neutral",        "Food Quality":
```

```
print(data_4.loc[7, 'model_response'])
```

```
    Sure! Here's the JSON output for the review you provided:{    "Overall": "Neutral",    "Food Quality": "Mixed
```

```
upd_val_1 = {
    "Overall": "Positive",
    "Food Quality": "Positive",
    "Service": "Positive",
    "Ambience": "Not Applicable",
    "Food Quality Features": [],
    "Service Features": ["excellent service"],
    "Ambience Features": []
}

upd_val_2 = {
    "Overall": "Neutral",
    "Food Quality": "Neutral",
    "Service": "Neutral",
    "Ambience": "Not Applicable",
    "Food Quality Features": ["well prepared"],
    "Service Features": ["slow and inattentive"],
    "Ambience Features": ["interior is friendly", "not intimidating"]
}

upd_val_3 = {
    "Overall": "Neutral",
    "Food Quality": "Positive",
    "Service": "Negative",
    "Ambience": "Positive",
    "Food Quality Features": ["Some tasty, others average"],
    "Service Features": ["Attentive staff", "Slow service"],
    "Ambience Features": []
}
idx_list = [3,6,7]
data_4.loc[idx_list, 'model_response_parsed'] = [upd_val_1, upd_val_2, upd_val_3]
```

```
model_response_parsed_df_4 = pd.json_normalize(data_4['model_response_parsed'])
model_response_parsed_df_4.head()
```

	Overall	Food Quality	Service	Ambience	Food Quality Features	Service Features	Ambience Features
0	Positive	Positive	Positive	Positive	[straight from the oven]	[staff wearing masks]	[quite fancy]
1	Positive	Positive	Positive	Neutral	[exquisite taste]	[freshly made]	[]
2	Positive	Positive	Positive	Positive	[Excellent taste]	[Great service]	[Awesome decorum]
3	Positive	Positive	Positive	Not Applicable	[]	[excellent service]	[]
4	Positive	Positive	Positive	Neutral	[fabulous]	[professional]	[]

Next steps: Generate code with model_response_parsed_df_4 New interactive sheet

```
data_with_parsed_model_output_4 = pd.concat([data_4, model_response_parsed_df_4], axis=1)
data_with_parsed_model_output_4.head()
```

	restaurant_ID	rating_review	review_full	model_response	model_response_parsed	Overall	Food Quality	Service	Am
0	FLV202	5	Totally in love with the Auro of the place, re...	{ "Overall": "Positive", "Fo...	{'Overall': 'Positive', 'Food Quality': 'Posit...	Positive	Positive	Positive	
1	SAV303	5	Kailash colony is brimming with small cafes no...	Sure! Here's my analysis of your restauran...	{'Overall': 'Positive', 'Food Quality': 'Posit...	Positive	Positive	Positive	
2	YUM789	5	Excellent taste and awesome decorum. Must visi...	Sure, I can assist you in that! Here's my ana...	{'Overall': 'Positive', 'Food Quality': 'Posit...	Positive	Positive	Positive	
3	TST101	5	I have visited at jw lough/restourant. There w...	Sure! Here's the JSON output based on your re...	{'Overall': 'Positive', 'Food Quality': 'Posit...	Positive	Positive	Positive	A
4	EAT456	5	Had a great experience in the restaurant food ...	{ "Overall": "Positive", "Fo...	{'Overall': 'Positive', 'Food Quality': 'Posit...	Positive	Positive	Positive	

Next steps: Generate code with data_with_parsed_model_output_4 New interactive sheet

```
final_data_4 = data_with_parsed_model_output_4.drop(['model_response','model_response_parsed'], axis=1)
final_data_4.head()
```

	restaurant_ID	rating_review	review_full	Overall	Food Quality	Service	Ambience	Food Quality Features	Service Features	Ambience Features
0	FLV202	5	Totally in love with the Auro of the place, re...	Positive	Positive	Positive	Positive	[straight from the oven]	[staff wearing masks]	[quite fancy]
1	SAV303	5	Kailash colony is brimming with small cafes no...	Positive	Positive	Positive	Neutral	[exquisite taste]	[freshly made]	[]
2	YUM789	5	Excellent taste and awesome decorum. Must visi...	Positive	Positive	Positive	Positive	[Excellent taste]	[Great service]	[Awesome decorum]

Next steps: Generate code with final_data_4 New interactive sheet

```
final_data_4['Overall'].value_counts()
```

Overall	count
Neutral	7
Positive	6
Negative	6
your_sentiment_prediction	1

dtype: int64

```
final_data_4['Food Quality'].value_counts()
```

Food Quality	count
Positive	8
Neutral	7
Not Applicable	2
Mixed	1
your_sentiment_prediction	1
Negative	1

dtype: int64

```
final_data_4['Service'].value_counts()
```

Service	count
Negative	9
Positive	8
Neutral	1
Slow	1
your_sentiment_prediction	1

dtype: int64

```
final_data_4['Ambience'].value_counts()
```

	count
Ambience	
Positive	6
Neutral	6
Not Applicable	6
your_sentiment_prediction	1
Negative	1

dtype: int64

```
data_5 = data.copy()
```

```
instruction_5 = """
    You are an AI analyzing restaurant reviews. Classify the overall sentiment of the provided review into the f
    - "Positive"
    - "Negative"
    - "Neutral"

    Once that is done, check for a mention of the following aspects in the review and clasify the sentiment of e
    1. Food quality
    2. Service
    3. Ambience

    Once that is done, look for liked and/or disliked features mentioned against each of the above aspects in th

    Finally, draft a response for the customer based on the review. Start out with a thank you note and then add
    1. If the review is positive, mention that it would be great to have them again
    2. If the review is neutral, ask them for what the restaurant could have done better
    3. If the review is negative, apologive for the inconvenience and mention that we'll be looking into the poi

    Return the output in the specified JSON format, ensuring consistency and handling missing values appropriate

    {
        "Overall": "your_sentiment_prediction",
        "Food Quality": "your_sentiment_prediction",
        "Service": "your_sentiment_prediction",
        "Ambience": "your_sentiment_prediction",
        "Food Quality Features": ["liked/disliked features"],
        "Service Features": ["liked/disliked features"],
        "Ambience Features": ["liked/disliked features"],
        "Response": "your_response_to_the_customer_review",
    }

    The sentiment prediction for Overall, Food Quality, Service, and Ambience should be one of "Positive", "Nega
    In case one of the three aspects is not mentioned in the review, set "Not Applicable" (including quotes) in
    In case there are no liked/disliked features for a particular aspect, assign an empty list in the correspond
    Be polite and empathetic in the response to the customer review.
    Only return the JSON, do NOT return any other text or information.
"""
```

```
data_5['model_response'] = data_5['review_full'].apply(lambda x: generate_llama_response(instruction_5, x))
```

```
llama_print_timings:      sample time =     298.84 ms /   472 runs   (    0.63 ms per token,  1579.42 tokens pe
llama_print_timings: prompt eval time =    1862.47 ms /   875 tokens (    2.13 ms per token,   469.81 tokens pe
llama_print_timings:        eval time =   32099.31 ms /   471 runs   (   68.15 ms per token,    14.67 tokens pe
llama_print_timings:       total time =   36141.45 ms /  1346 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     185.74 ms /   311 runs   (    0.60 ms per token,  1674.39 tokens pe
llama_print_timings: prompt eval time =    1881.90 ms /   879 tokens (    2.14 ms per token,   467.08 tokens pe
llama_print_timings:        eval time =   21071.08 ms /   310 runs   (   67.97 ms per token,    14.71 tokens pe
llama_print_timings:       total time =   24370.95 ms /  1189 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     261.85 ms /   423 runs   (    0.62 ms per token,  1615.42 tokens pe
llama_print_timings: prompt eval time =    2007.16 ms /   943 tokens (    2.13 ms per token,   469.82 tokens pe
llama_print_timings:        eval time =   28905.55 ms /   422 runs   (   68.50 ms per token,    14.60 tokens pe
llama_print_timings:       total time =   32836.22 ms /  1365 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =     153.09 ms /   271 runs   (    0.56 ms per token,  1770.19 tokens pe
llama_print_timings: prompt eval time =    2481.38 ms /  1073 tokens (    2.31 ms per token,   432.42 tokens pe
llama_print_timings:        eval time =   18741.51 ms /   270 runs   (   69.41 ms per token,    14.41 tokens pe
llama_print_timings:       total time =   22367.51 ms /  1343 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1033.19 ms
llama_print_timings:      sample time =      97.58 ms /   156 runs   (    0.63 ms per token,  1598.64 tokens pe
llama_print_timings: prompt eval time =    1911.83 ms /   896 tokens (    2.13 ms per token,   468.66 tokens pe
llama_print_timings:        eval time =   10470.89 ms /   155 runs   (   67.55 ms per token,    14.80 tokens pe
llama_print_timings:       total time =   13057.34 ms /  1051 tokens
```

```python
i = 2
print(data_5.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had given us a great service. One of the best exp
```

```python
print(data_5.loc[i, 'model_response'])
```

```
 {
"Overall": "Positive",
"Food Quality": "Positive",
"Service": "Positive",
"Ambience": "Positive",
"Food Quality Features": ["Excellent taste"],
"Service Features": ["Great service"],
"Ambience Features": ["Awesome decorum"],
"Response": "Thank you for taking the time to share your positive review with us! We're thrilled to hear that you
}
```

```python
data_5['model_response_parsed'] = data_5['model_response'].apply(extract_json_data)
data_5['model_response_parsed'].head()
```

```
Error parsing JSON: Expecting property name enclosed in double quotes: line 10 column 5 (char 539)
Error parsing JSON: Expecting ',' delimiter: line 9 column 277 (char 503)
Error parsing JSON: Expecting property name enclosed in double quotes: line 10 column 5 (char 579)
Error parsing JSON: Expecting property name enclosed in double quotes: line 10 column 5 (char 669)
```

	model_response_parsed
0	{'Overall': 'Positive', 'Food Quality': 'Posit...
1	{}
2	{}
3	{'Overall': 'Positive', 'Food Quality': 'Posit...
4	{}

dtype: object

```python
model_response_parsed_df_5 = pd.json_normalize(data_5['model_response_parsed'])
model_response_parsed_df_5.head()
```

	Overall	Food Quality	Service	Ambience	Food Quality Features	Service Features	Ambience Features	Response
0	Positive	Positive	Positive	Positive	[straight from the oven]	[quite delicious]	[quite quaint and cute]	Thank you for taking the time to review us! We...
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Positive	Positive	Positive	Positive	[superb food]	[superb service]	[very nice ambience]	Thank you so much for taking the time to revie...

Next steps: Generate code with model_response_parsed_df_5 New interactive sheet

```
model_response_parsed_df_5['Response'][0]
```

```
'Thank you for taking the time to review us! We're thrilled to hear that you enjoyed our food quality, service,
and ambience. We're especially glad that you appreciated our open kitchen concept and our use of disposable cutl
ery to ensure your safety during these challenging times. We hope to have you back again soon! If you have any f
urther suggestions or feedback, please don't hesitate to reach out to us.'
```

```
data_with_parsed_model_output_5 = pd.concat([data_5, model_response_parsed_df_5], axis=1)
data_with_parsed_model_output_5.head()
```

	restaurant_ID	rating_review	review_full	model_response	model_response_parsed	Overall	Food Quality	Service	Am
0	FLV202	5	Totally in love with the Auro of the place, re...	\n {\n "Overall": "Positive",\n ...	{'Overall': 'Positive', 'Food Quality': 'Posit...	Positive	Positive	Positive	
1	SAV303	5	Kailash colony is brimming with small cafes no...	\n {\n "Overall": "Positive",\n ...	{}	NaN	NaN	NaN	
2	YUM789	5	Excellent taste and awesome decorum. Must visi...	{\n"Overall": "Positive",\n"Food Quality": "P...	{}	NaN	NaN	NaN	
3	TST101	5	I have visited at jw lough/restourant. There w...	{\n "Overall": "Positive",\n "F...	{'Overall': 'Positive', 'Food Quality': 'Posit...	Positive	Positive	Positive	
4	EAT456	5	Had a great experience in the restaurant food ...	\n {\n "Overall": "Positive",\n ...	{}	NaN	NaN	NaN	

Next steps: Generate code with data_with_parsed_model_output_5 New interactive sheet

```
final_data_5 = data_with_parsed_model_output_5.drop(['model_response','model_response_parsed'], axis=1)
final_data_5.head()
```

	restaurant_ID	rating_review	review_full	Overall	Food Quality	Service	Ambience	Food Quality Features	Service Features	Ambience Features	R
0	FLV202	5	Totally in love with the Auro of the place, re...	Positive	Positive	Positive	Positive	[straight from the oven]	[quite delicious]	[quite quaint and cute]	T th r
1	SAV303	5	Kailash colony is brimming with small cafes no...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	YUM789	5	Excellent taste and awesome decorum. Must visi...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	TST101	5	I have visited at jw lough/restourant	Positive	Positive	Positive	Positive	[superb food]	[superb service]	[very nice ambience]	T

Next steps: Generate code with final_data_5 New interactive sheet

```
final_data_5['Overall'].value_counts()
```

```
         count
Overall
Neutral      9
Negative     5
Positive     2

dtype: int64
```

```
final_data_5['Food Quality'].value_counts()
```