Install Libraries

```
pip install sentence_transformers -q
```

```

!pip install --quiet bitsandbytes==0.41.1
```

```
pip install accelerate -q
```

```
pip install langchain_openai langchain_community langchain_experimental -q
```

```
pip install langchain huggingface_hub -q
```

```
pip install unstructured -q
```

```
pip install pydantic==2.6.0 -q
```

```
pip install lark chromadb faiss-gpu -q
```

```
!pip install optimum -q
```

```
!nvidia-smi
```

```
Tue Apr 16 17:59:22 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla V100-SXM2-16GB           Off | 00000000:00:04.0 Off |                    0 |
| N/A   34C    P0              25W / 300W |      0MiB / 16384MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
```

Import Libraries

```
import torch
from torch import cuda, bfloat16
import transformers
import pandas as pd

import sentence_transformers
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain.prompts import ChatPromptTemplate
from langchain.vectorstores import FAISS
from langchain_community.llms import HuggingFaceHub
from langchain_community.llms.huggingface_pipeline import HuggingFacePipeline
from langchain.prompts import PromptTemplate
from langchain_core.runnables import RunnableLambda, RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser
```

```
device = f'cuda:{cuda.current_device()}' if cuda.is_available() else 'cpu'
device
```

```
'cuda:0'
```

1. Read Data

```
path = "/content/drive/MyDrive/DATAGPT/take_home_dataset.csv"
```

```
df = pd.read_csv(path, delimiter=";")
```

```
columns_to_use = ['Date', 'Order_ID', 'Product_Category', 'Delivery_distance']
df = df[columns_to_use]
df.reset_index(inplace=True)
df.rename(columns={'index': 'Row_ID'}, inplace=True)
```

```
df.head()
```

	Row_ID	Date	Order_ID	Product_Category	Delivery_distance
0	0	2023-07-01	3808	Apparel	300
1	1	2023-07-02	3808	Apparel	300
2	2	2023-07-03	3808	Apparel	300
3	3	2023-07-04	3808	Apparel	300
4	4	2023-07-01	487	Cosmetics & Personal Care	500

Next steps: View recommended plots

Preprocessing

```
metadata = {
    'Row_ID' : 'numeric',
    'Date' : "datetime",
    'Order_ID' : 'numeric',
    'Product_Category' : 'categorical',
    'Delivery_distance' : 'numeric'
}
```

Create Embedding Model

```
embed_model_id = 'sentence-transformers/all-MiniLM-L6-v2'

embed_model = HuggingFaceEmbeddings(
    model_name=embed_model_id,
    model_kwargs={'device': device},
    encode_kwargs={'device': device, 'batch_size': 32}
)
```

```
/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://huggingface.co/settings/tokens), set it as secre
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(
```

3. Embedding Model

- create column embeddings
- find column_column name from query
- get column type from meatadata
- if type == categorical: get value from column similar to query

- elif type in [datetime, numeric]: extract exact value from query

Column Embeddings to get column name

```
column_embeddings = FAISS.from_texts(columns_to_use, embed_model)
```

```
def get_column_name(query, embeddings):
    return embeddings.similarity_search(query, k = 1)[0].page_content
```

```
def get_column_type(column, metadata):
    return metadata[column]
```

```
def extract_similar_value(query, data, column, embed_model):
    '''
    This method is executed when the column is categorical
    '''
    col_values_embedding = FAISS.from_texts(data[column].unique().astype(str), embed_model)
    extracted_value = col_values_embedding.similarity_search(query, k = 1)[0].page_content
    return extracted_value
```

Rephrase query if column is categorical

```
def get_final_query(query, embed_model, column_embeddings, data, metadata):
    similar_column = get_column_name(query, column_embeddings)
    column_type = get_column_type(similar_column, metadata)
    if column_type == 'categorical':
        extracted_column_value = extract_similar_value(query, data, similar_column, embed_model)
        final_query = similar_column + " " + extracted_column_value
        return final_query
    else:
        final_query = query
        return final_query
```

Model

```
from torch import cuda, bfloat16
import transformers

model_id = 'meta-llama/Llama-2-13b-chat-hf'


# set quantization configuration to load large model with less GPU memory
# this requires the `bitsandbytes` library
bnb_config = transformers.BitsAndBytesConfig(
    bnb_4bit_quant_type='nf4',
    load_in_8bit_fp32_cpu_offload=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=bfloat16
)

# begin initializing HF items, need auth token for these
hf_auth = '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'
model_config = transformers.AutoConfig.from_pretrained(
    model_id,
    use_auth_token=hf_auth
)

model = transformers.AutoModelForCausalLM.from_pretrained(
    model_id,
    trust_remote_code=True,
    config=model_config,
    quantization_config=bnb_config,
    device_map='auto',
    use_auth_token=hf_auth
)
model.eval()
print(f"Model loaded on {device}")

# tokentizer
tokenizer = transformers.AutoTokenizer.from_pretrained(
    model_id,
    use_auth_token=hf_auth
)

# pipeline
llm_pipeline = transformers.pipeline(
    model=model, tokenizer=tokenizer,
    return_full_text=True,  # langchain expects the full text
    task='text-generation',
    # we pass model parameters here too
    temperature=0.1,  # 'randomness' of outputs, 0.0 is the min and 1.0 the max
    max_new_tokens=1024,  # mex number of tokens to generate in the output
    repetition_penalty=1.1  # without this output begins repeating
)

```

```
/usr/local/lib/python3.10/dist-packages/transformers/models/auto/configuration_auto.py:1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/models/auto/auto_factory.py:466: Fu
  warnings.warn(
Loading checkpoint shards: 100%  3/3 [02:01<00:00, 39.01s/it]
You are calling `save_pretrained` to a 4-bit converted model, but your `bitsandbytes` ve
/usr/local/lib/python3.10/dist-packages/transformers/models/auto/tokenization_auto.py:72
  warnings.warn(
Model loaded on cuda:0
```

Parser Logic

```
llm = HuggingFacePipeline(pipeline=llm_pipeline)
```

```
pandas_prompt_template = """<<SYS>>
    You are a data analyst working on a pandas datframe 'df'.\n Use df.head for your reference.Your job is to return a Pandas expression
    [INST]
    Instructions: {instructions}
    df: {df_head}
    Query: {query}
    PRINT ONLY THE PANDAS EXPRESSION after Expression--> AT THE END OF RESPONSE.
    [/INST]
    END
"""
```

```
extraction_prompt = """<<SYS>>
    You are a data analyst working on a pandas datframe 'df'.\nYour job is to extract 'column_name' and 'value' from the pandas expressio
    [INST]
    PRINT ONLY THE `column_name` AND `value` AFTER  SOLUTION--> at the end of response.
    expression: {expression}
    [/INST]
    END
"""
```

```
instructions_prompt = """
1. RETURN THE PANDAS EXPRESSION ONLY.
Here are the possible columns:
```
[{columns}]
```
"""
```

```
prompt = ChatPromptTemplate.from_template(pandas_prompt_template)
```

```
extraction_prompt = ChatPromptTemplate.from_template(extraction_prompt)
```

```
final_prompt = prompt.partial(instructions = instructions_prompt)
```

```
chain = (
    {"query": RunnablePassthrough(), "columns": lambda x: list(df.columns), "df_head": lambda x:df.head(5).to_dict()}
    | final_prompt
    | llm
    | StrOutputParser()
)
```

```
extraction_chain =  (
    {"expression": RunnablePassthrough()}
    | extraction_prompt
    | llm
    | StrOutputParser()
)
```

fetch query from response

```
def extract_query_from_response(answer):
    pandas_query = answer.split("\n")[-1]
    return pandas_query
```

Get row indices satisfying query

```
def get_row_id(df, answer):
    pandas_query = extract_query_from_response(answer)
    result_df = eval(pandas_query)
    if isinstance(result_df, pd.DataFrame):
        indices = result_df.index.tolist()
    else:
        indices = df.index[result_df].tolist()
    return indices
```

Method to extract column_name and value

```
def extract_col_and_value(answer):
    pandas_query = extract_query_from_response(answer)
    extraction_chain_response = extraction_chain.invoke(pandas_query)
    column_name = extraction_chain_response.split("END")[1].strip().split("column_name:")[1].split("\n")[0].strip()
    value = extraction_chain_response.split("END")[1].strip().split("value:")[1].split("\n")[0].strip()
    return column_name, value
```

```
def generate_output(answer, df):
    pandas_query = extract_query_from_response(answer)
    column_name, value = extract_col_and_value(answer)
    indices = get_row_id(df, answer)
    return {"column_name": column_name, "value": value, "row_ids" : indices}
```

Runner

```
def run(query):
    final_query = get_final_query(query, embed_model, column_embeddings, df, metadata)
    answer = chain.invoke(final_query)
    pandas_query = extract_query_from_response(answer)
    indices = get_row_id(df, answer)
    output = generate_output(answer, df)
    print(output)

```

```
run("delivery distance greater then 500")
```

```
{'column_name': 'Delivery_distance', 'value': '> 500', 'row_ids': [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 3
```

```
run("product apparel")
```

```
{'column_name': 'Product_Category', 'value': 'Apparel', 'row_ids': [0, 1, 2, 3, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
```

```
run("product machine")
```

```
{'column_name': 'Product_Category', 'value': 'Electronics', 'row_ids': [89, 90, 91, 92, 93, 94, 95, 96]}
```

```
run("product toys")
```

```
{'column_name': 'Product_Category', 'value': 'Toys & Games', 'row_ids': [39, 40, 41, 42, 43, 44, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
```

```

run("orders between 1 july 2023 and 5 july 2023")
```

```
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1157: UserWarning: You seem to be using the pipelines sequentiall
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1157: UserWarning: You seem to be using the pipelines sequentiall
  warnings.warn(
{'column_name': 'Date', 'value': '2023-07-01', 'row_ids': [1, 2, 3, 5, 6, 7, 10, 11, 12, 15, 16, 17, 29, 30, 31, 34, 35, 36, 40, 41, 42,
```

```
run("order id more than 4000")
```

```
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1157: UserWarning: You seem to be using the pipelines sequentiall
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1157: UserWarning: You seem to be using the pipelines sequentiall
  warnings.warn(
{'column_name': 'Order_ID', 'value': '4001', 'row_ids': [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
```

```
run("delivery distance greater then 500")
```

```
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1157: UserWarning: You seem to be using the pipelines sequentiall
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1157: UserWarning: You seem to be using the pipelines sequentiall
```

```
  warnings.warn(
{'column_name': 'Delivery_distance', 'value': '> 500', 'row_ids': [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 3
```

```
1
```