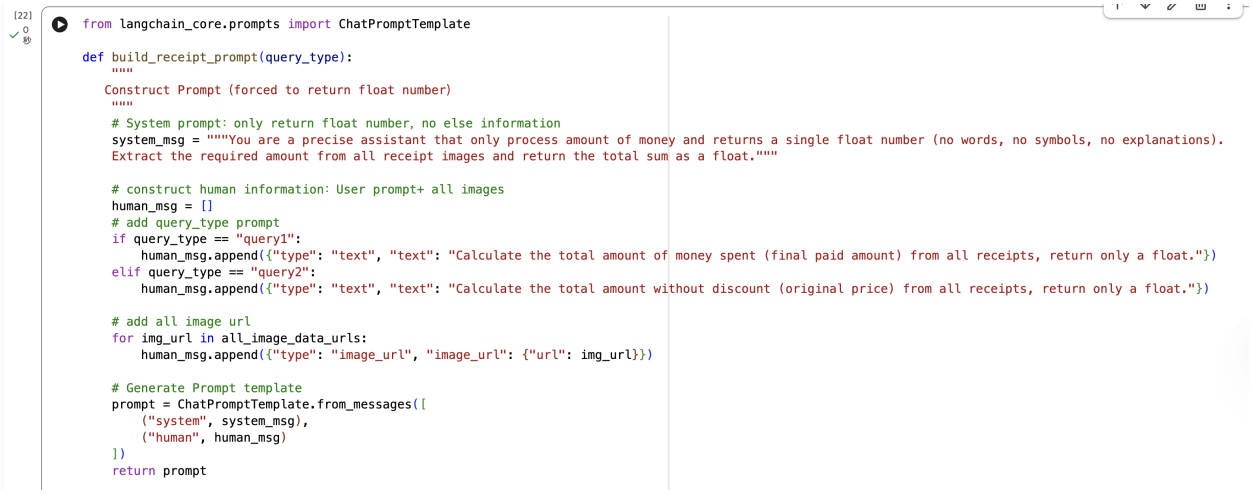
1. First define a function to reject irrelevant queries and return accurate query type.

```
[21]
0 秒
#Generate Data URL
all_image_data_urls = [get_image_data_url(path) for path in image_paths]

# Define the core query function(reject irrelevant information)
def classify_query(user_query):
    """
    classify user query

    return "query1"/"query2"/"irrelevant"
    """
    # Formalize query (lower+strip)
    query_clean = user_query.lower().strip()

    # Query1 key words
    query1_keywords = ["how much money did i spend in total", "total spent", "spend in total" ]
    # Query2 key words
    query2_keywords = ["without the discount", "no discount","pay without the discount"]

    # Query1
    if any(keyword in query_clean for keyword in query1_keywords):
        return "query1"
    # Query2
    elif any(keyword in query_clean for keyword in query2_keywords):
        return "query2"
    # Irrelevant
    else:
        return "irrelevant"
```

2. Secondly build the prompt for the model,include system prompt and user prompt. Define the system prompt :only return float number of the results



```
[22]
0 秒
from langchain_core.prompts import ChatPromptTemplate

def build_receipt_prompt(query_type):
    """
   Construct Prompt (forced to return float number)
    """
    # System prompt: only return float number, no else information
    system_msg = """You are a precise assistant that only process amount of money and returns a single float number (no words, no symbols, no explanations).
    Extract the required amount from all receipt images and return the total sum as a float."""

    # construct human information: User prompt+ all images
    human_msg = []
    # add query_type prompt
    if query_type == "query1":
        human_msg.append({"type": "text", "text": "Calculate the total amount of money spent (final paid amount) from all receipts, return only a float."})
    elif query_type == "query2":
        human_msg.append({"type": "text", "text": "Calculate the total amount without discount (original price) from all receipts, return only a float."})

    # add all image url
    for img_url in all_image_data_urls:
        human_msg.append({"type": "image_url", "image_url": {"url": img_url}})

    # Generate Prompt template
    prompt = ChatPromptTemplate.from_messages([
        ("system", system_msg),
        ("human", human_msg)
    ])
    return prompt
```

3. Thirdly, define the main function to process the main query, build prompt and invoke the chain response, use regular expression to ensure the answer only returns a float number

```python
import re
def process_receipt_query(user_query):
    """
    Process the main query:
    - irrelevant query: return "Rejected: Irrelevant query"
    - core query: return float number
    """
    # define query type
    query_type = classify_query(user_query)

    # reject the irrelevant
    if query_type == "irrelevant":
        return "Rejected: Irrelevant query"

    # process the main query
    prompt = build_receipt_prompt(query_type)
    chain = prompt | llm
    response = chain.invoke({})
    content = response.content

    if isinstance(content, list):
        raw_output = "".join([str(item) for item in content]).strip()
    elif isinstance(content, str):
        raw_output = content.strip()

    # extract float number (Regular Expression)
    try:
        # pair with number
        numbers = re.findall(r'\d+\.\d+|\d+', raw_output)
        if not numbers:
            raise ValueError("No valid number found in model output")
        # get the last number(default)
        total_amount = float(numbers[-1])
        return total_amount
    except Exception as e:
        raise ValueError(f"Failed to parse model output: {raw_output}, Error: {e}")
```

4. Test results ,I think the model is not strong enough to find the discount amount of number and I changed the model gemini-2.5-pro but also failed to match the number.

Run the following code block to evaluate query 1:

> How much money did I spend in total for these bills?

[31] 10秒

```python
query_1_costs = [394.7, 316.1, 140.8, 514.0, 102.3, 190.8, 315.6] # do not modify this
query1_answer = process_receipt_query(user_query1)
print(query1_answer)
test_query(query1_answer, query_1_costs)
```

```
1974.3
```

Run the following code block to evaluate query 2:

> How much would I have had to pay without the discount?

[40] 1分钟

```python
query_2_costs = [480.20, 392.20, 160.10, 590.80, 107.70, 221.20, 396.00] # do not modify this
query2_answer = process_receipt_query(user_query2)
print(query2_answer)#Maybe the model is not capable enough to solve this problem.
#test_query(query2_answer, query_2_costs)
```

```
2301.77
```

[ ]

```python
sum([480.20, 392.20, 160.10, 590.80, 107.70, 221.20, 396.00])
```

```
2348.2
```

[34] 0秒

```python
query3_answer = process_receipt_query(user_query3)
print(query3_answer)
```

```
Rejected: Irrelevant query
```