AI ASSIST CODING

M JAGADEESH

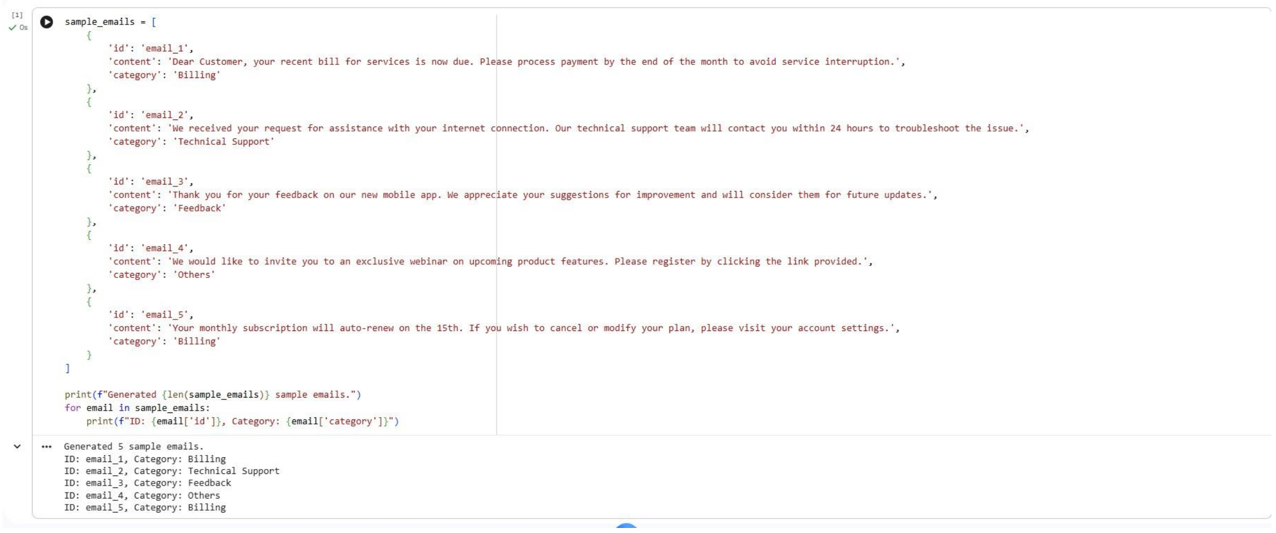
2303A52290

ASS 4.1

1) Customer Email Classification

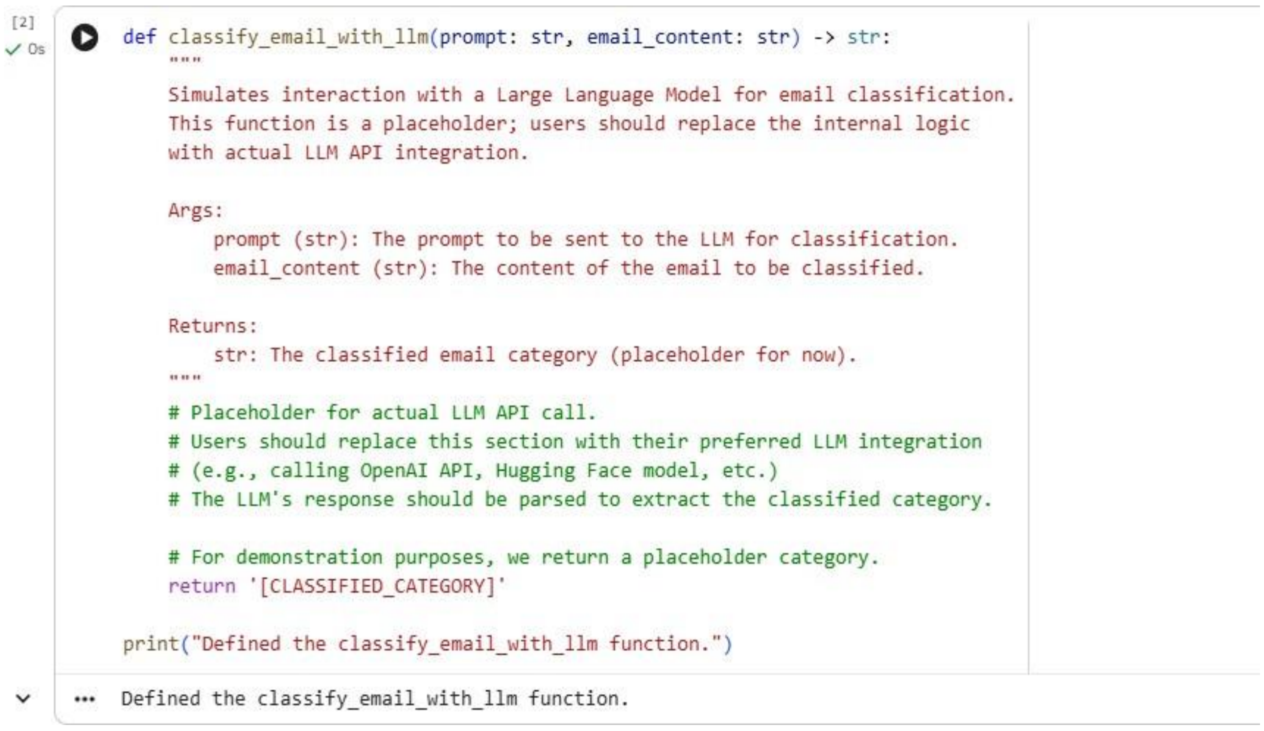
Prompt: Give me the code He company decides to use prompt engineering techniques with an existing large language model.

Code:



```
[1]
✓ 0s
sample_emails = [
    {
        'id': 'email_1',
        'content': 'Dear Customer, your recent bill for services is now due. Please process payment by the end of the month to avoid service interruption.',
        'category': 'Billing'
    },
    {
        'id': 'email_2',
        'content': 'We received your request for assistance with your internet connection. Our technical support team will contact you within 24 hours to troubleshoot the issue.',
        'category': 'Technical Support'
    },
    {
        'id': 'email_3',
        'content': 'Thank you for your feedback on our new mobile app. We appreciate your suggestions for improvement and will consider them for future updates.',
        'category': 'Feedback'
    },
    {
        'id': 'email_4',
        'content': 'We would like to invite you to an exclusive webinar on upcoming product features. Please register by clicking the link provided.',
        'category': 'Others'
    },
    {
        'id': 'email_5',
        'content': 'Your monthly subscription will auto-renew on the 15th. If you wish to cancel or modify your plan, please visit your account settings.',
        'category': 'Billing'
    }
]

print(f"Generated {len(sample_emails)} sample emails.")
for email in sample_emails:
    print(f"ID: {email['id']}, Category: {email['category']}")
```

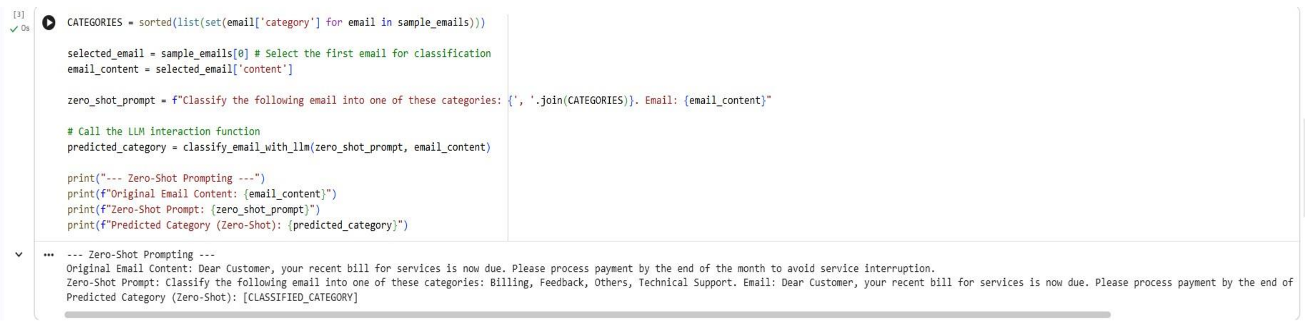
```
Generated 5 sample emails.
ID: email_1, Category: Billing
ID: email_2, Category: Technical Support
ID: email_3, Category: Feedback
ID: email_4, Category: Others
ID: email_5, Category: Billing
```



[2]
✓ 0s

```python
def classify_email_with_llm(prompt: str, email_content: str) -> str:
    """
    Simulates interaction with a Large Language Model for email classification.
    This function is a placeholder; users should replace the internal logic
    with actual LLM API integration.

    Args:
        prompt (str): The prompt to be sent to the LLM for classification.
        email_content (str): The content of the email to be classified.

    Returns:
        str: The classified email category (placeholder for now).
    """
    # Placeholder for actual LLM API call.
    # Users should replace this section with their preferred LLM integration
    # (e.g., calling OpenAI API, Hugging Face model, etc.)
    # The LLM's response should be parsed to extract the classified category.

    # For demonstration purposes, we return a placeholder category.
    return '[CLASSIFIED_CATEGORY]'

print("Defined the classify_email_with_llm function.")
```

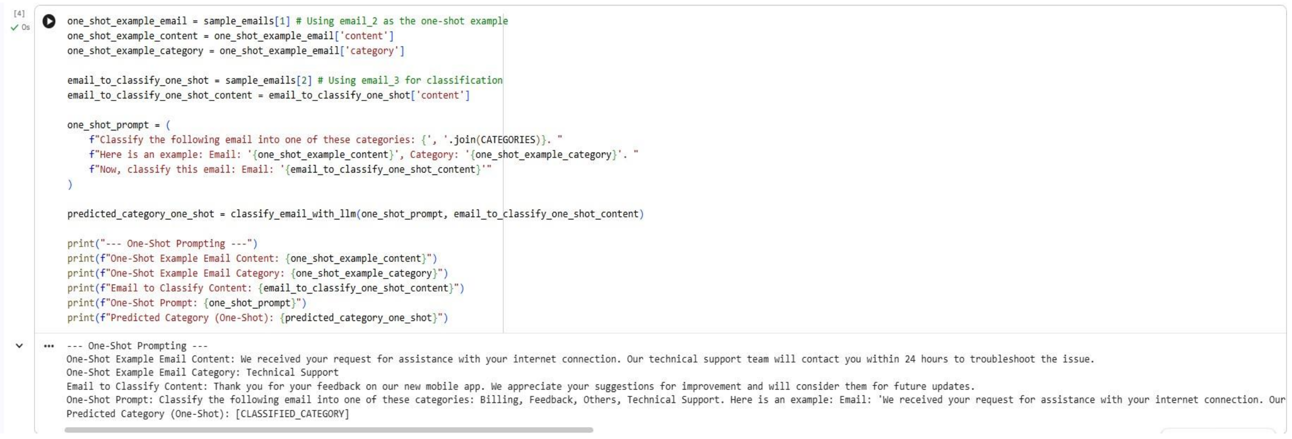
```
Defined the classify_email_with_llm function.
```



[3]
✓ 0s

```python
CATEGORIES = sorted(list(set(email['category'] for email in sample_emails)))

selected_email = sample_emails[0] # Select the first email for classification
email_content = selected_email['content']

zero_shot_prompt = f"Classify the following email into one of these categories: {', '.join(CATEGORIES)}. Email: {email_content}"

# Call the LLM interaction function
predicted_category = classify_email_with_llm(zero_shot_prompt, email_content)

print("--- Zero-Shot Prompting ---")
print(f"Original Email Content: {email_content}")
print(f"Zero-Shot Prompt: {zero_shot_prompt}")
print(f"Predicted Category (Zero-Shot): {predicted_category}")
```

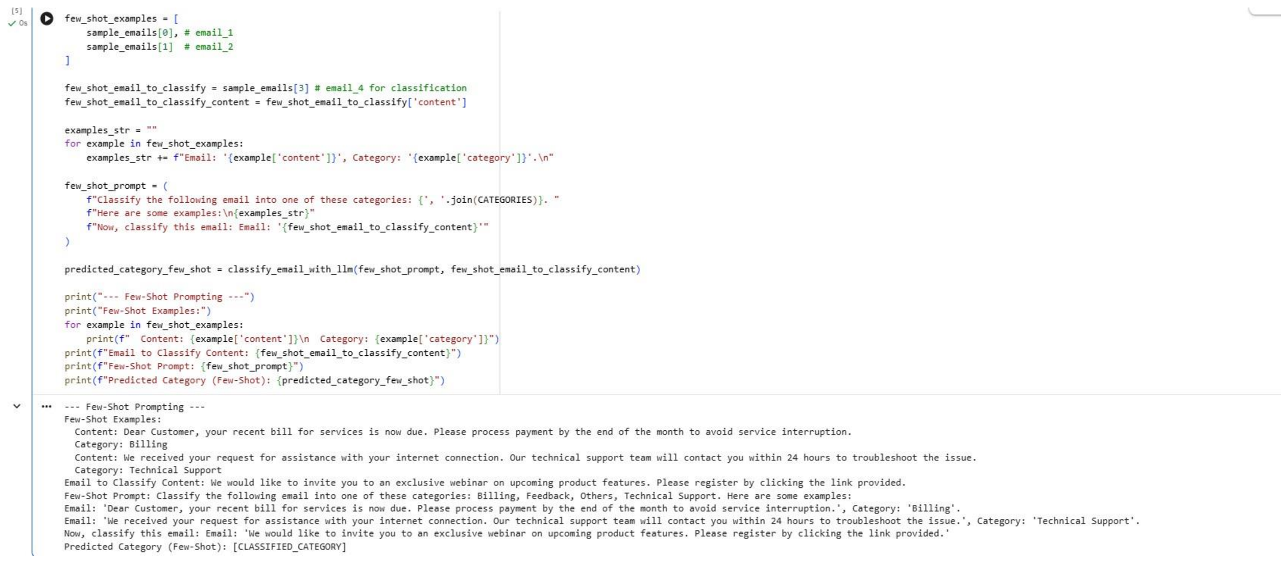
```
--- Zero-Shot Prompting ---
Original Email Content: Dear Customer, your recent bill for services is now due. Please process payment by the end of the month to avoid service interruption.
Zero-Shot Prompt: Classify the following email into one of these categories: Billing, Feedback, Others, Technical Support. Email: Dear Customer, your recent bill for services is now due. Please process payment by the end of
Predicted Category (Zero-Shot): [CLASSIFIED_CATEGORY]
```



[4]
✓ 0s

```python
one_shot_example_email = sample_emails[1] # Using email_2 as the one-shot example
one_shot_example_content = one_shot_example_email['content']
one_shot_example_category = one_shot_example_email['category']

email_to_classify_one_shot = sample_emails[2] # Using email_3 for classification
email_to_classify_one_shot_content = email_to_classify_one_shot['content']

one_shot_prompt = (
    f"Classify the following email into one of these categories: {', '.join(CATEGORIES)}. "
    f"Here is an example: Email: '{one_shot_example_content}', Category: '{one_shot_example_category}'. "
    f"Now, classify this email: Email: '{email_to_classify_one_shot_content}'"
)

predicted_category_one_shot = classify_email_with_llm(one_shot_prompt, email_to_classify_one_shot_content)

print("--- One-Shot Prompting ---")
print(f"One-Shot Example Email Content: {one_shot_example_content}")
print(f"One-Shot Example Email Category: {one_shot_example_category}")
print(f"Email to Classify Content: {email_to_classify_one_shot_content}")
print(f"One-Shot Prompt: {one_shot_prompt}")
print(f"Predicted Category (One-Shot): {predicted_category_one_shot}")
```

```
--- One-Shot Prompting ---
One-Shot Example Email Content: We received your request for assistance with your internet connection. Our technical support team will contact you within 24 hours to troubleshoot the issue.
One-Shot Example Email Category: Technical Support
Email to Classify Content: Thank you for your feedback on our new mobile app. We appreciate your suggestions for improvement and will consider them for future updates.
One-Shot Prompt: Classify the following email into one of these categories: Billing, Feedback, Others, Technical Support. Here is an example: Email: 'We received your request for assistance with your internet connection. Our
Predicted Category (One-Shot): [CLASSIFIED_CATEGORY]
```

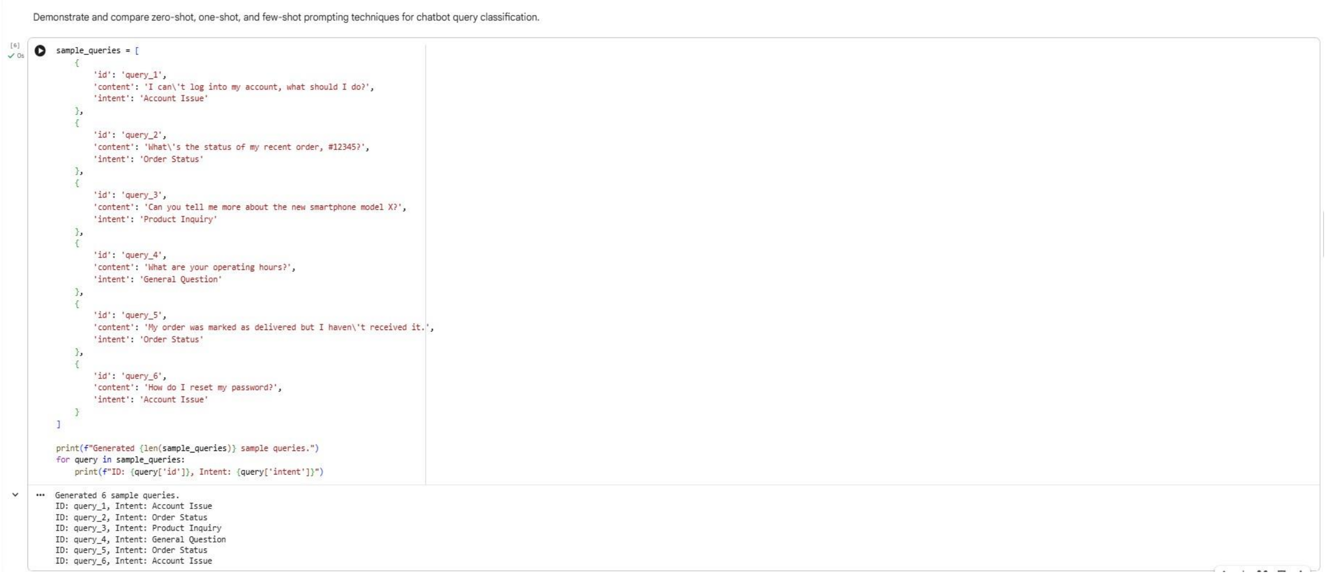


```
few_shot_examples = [
    sample_emails[0], # email_1
    sample_emails[1]  # email_2
]

few_shot_email_to_classify = sample_emails[3] # email_4 for classification
few_shot_email_to_classify_content = few_shot_email_to_classify['content']

examples_str = ""
for example in few_shot_examples:
    examples_str += f"Email: '{example['content']}', Category: '{example['category']}'.\n"

few_shot_prompt = (
    f"Classify the following email into one of these categories: {', '.join(CATEGORIES)}. "
    f"Here are some examples:\n{examples_str}"
    f"Now, classify this email: Email: '{few_shot_email_to_classify_content}'"
)

predicted_category_few_shot = classify_email_with_llm(few_shot_prompt, few_shot_email_to_classify_content)

print("--- Few-Shot Prompting ---")
print("Few-Shot Examples:")
for example in few_shot_examples:
    print(f"  Content: {example['content']}\n  Category: {example['category']}")
print(f"Email to Classify Content: {few_shot_email_to_classify_content}")
print(f"Few-Shot Prompt: {few_shot_prompt}")
print(f"Predicted Category (Few-Shot): {predicted_category_few_shot}")
```

```
--- Few-Shot Prompting ---
Few-Shot Examples:
  Content: Dear Customer, your recent bill for services is now due. Please process payment by the end of the month to avoid service interruption.
  Category: Billing
  Content: We received your request for assistance with your internet connection. Our technical support team will contact you within 24 hours to troubleshoot the issue.
  Category: Technical Support
Email to Classify Content: We would like to invite you to an exclusive webinar on upcoming product features. Please register by clicking the link provided.
Few-Shot Prompt: Classify the following email into one of these categories: Billing, Feedback, Others, Technical Support. Here are some examples:
Email: 'Dear Customer, your recent bill for services is now due. Please process payment by the end of the month to avoid service interruption.', Category: 'Billing'.
Email: 'We received your request for assistance with your internet connection. Our technical support team will contact you within 24 hours to troubleshoot the issue.', Category: 'Technical Support'.
Now, classify this email: Email: 'We would like to invite you to an exclusive webinar on upcoming product features. Please register by clicking the link provided.'
Predicted Category (Few-Shot): [CLASSIFIED_CATEGORY]
```

2) Intent Classification for Chatbot Queries

Prompt: Account Issue, Order Status, Product Inquiry, or General Question using prompt engineering techniques.



Demonstrate and compare zero-shot, one-shot, and few-shot prompting techniques for chatbot query classification.

```
sample_queries = [
    {
        'id': 'query_1',
        'content': 'I can\'t log into my account, what should I do?',
        'intent': 'Account Issue'
    },
    {
        'id': 'query_2',
        'content': 'What\'s the status of my recent order, #12345?',
        'intent': 'Order Status'
    },
    {
        'id': 'query_3',
        'content': 'Can you tell me more about the new smartphone model X?',
        'intent': 'Product Inquiry'
    },
    {
        'id': 'query_4',
        'content': 'What are your operating hours?',
        'intent': 'General Question'
    },
    {
        'id': 'query_5',
        'content': 'My order was marked as delivered but I haven\'t received it.',
        'intent': 'Order Status'
    },
    {
        'id': 'query_6',
        'content': 'How do I reset my password?',
        'intent': 'Account Issue'
    }
]

print(f"Generated {len(sample_queries)} sample queries.")
for query in sample_queries:
    print(f"ID: {query['id']}, Intent: {query['intent']}")
```

```
Generated 6 sample queries.
ID: query_1, Intent: Account Issue
ID: query_2, Intent: Order Status
ID: query_3, Intent: Product Inquiry
ID: query_4, Intent: General Question
ID: query_5, Intent: Order Status
ID: query_6, Intent: Account Issue
```

[7]
✓ 0s

```python
    def classify_query_with_llm(prompt: str, query_content: str) -> str:
        """
        Simulates interaction with a Large Language Model for chatbot query classification.
        This function is a placeholder; users should replace the internal logic
        with actual LLM API integration.

        Args:
            prompt (str): The prompt to be sent to the LLM for classification.
            query_content (str): The content of the chatbot query to be classified.

        Returns:
            str: The classified query intent (placeholder for now).
        """
        # Placeholder for actual LLM API call.
        # Users should replace this section with their preferred LLM integration
        # (e.g., calling OpenAI API, Hugging Face model, etc.)
        # The LLM's response should be parsed to extract the classified intent.

        # For demonstration purposes, we return a placeholder intent.
        return '[CLASSIFIED_INTENT]'

    print("Defined the classify_query_with_llm function.")
```

```
Defined the classify_query_with_llm function.
```

[8]
✓ 0s

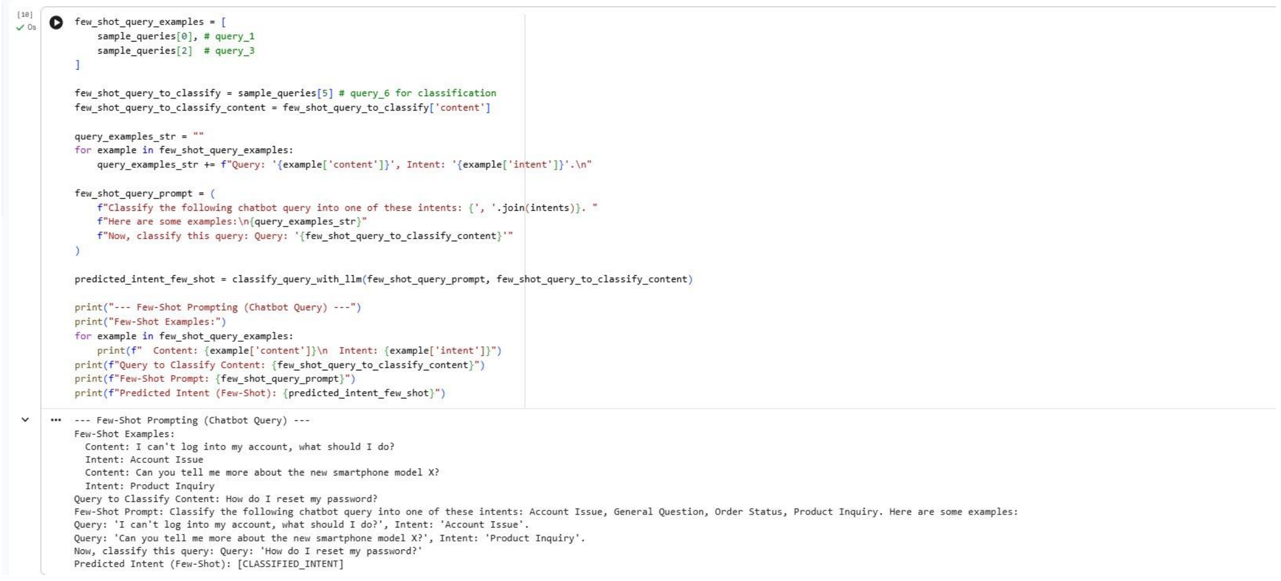
```python
intents = sorted(list(set(query['intent'] for query in sample_queries)))

selected_query = sample_queries[0] # Select the first query for classification
query_content = selected_query['content']

zero_shot_query_prompt = f"Classify the following chatbot query into one of these intents: {', '.join(intents)}. Query: {query_content}"

# Call the LLM interaction function
predicted_intent = classify_query_with_llm(zero_shot_query_prompt, query_content)

print("--- Zero-Shot Prompting (Chatbot Query) ---")
print(f"Original Query Content: {query_content}")
print(f"Zero-Shot Prompt: {zero_shot_query_prompt}")
print(f"Predicted Intent (Zero-Shot): {predicted_intent}")
```

```
--- Zero-Shot Prompting (Chatbot Query) ---
Original Query Content: I can't log into my account, what should I do?
Zero-Shot Prompt: Classify the following chatbot query into one of these intents: Account Issue, General Question, Order Status, Product Inquiry. Query: I can't log into my account, what should I
Predicted Intent (Zero-Shot): [CLASSIFIED_INTENT]
```

[9]
✓ 0s

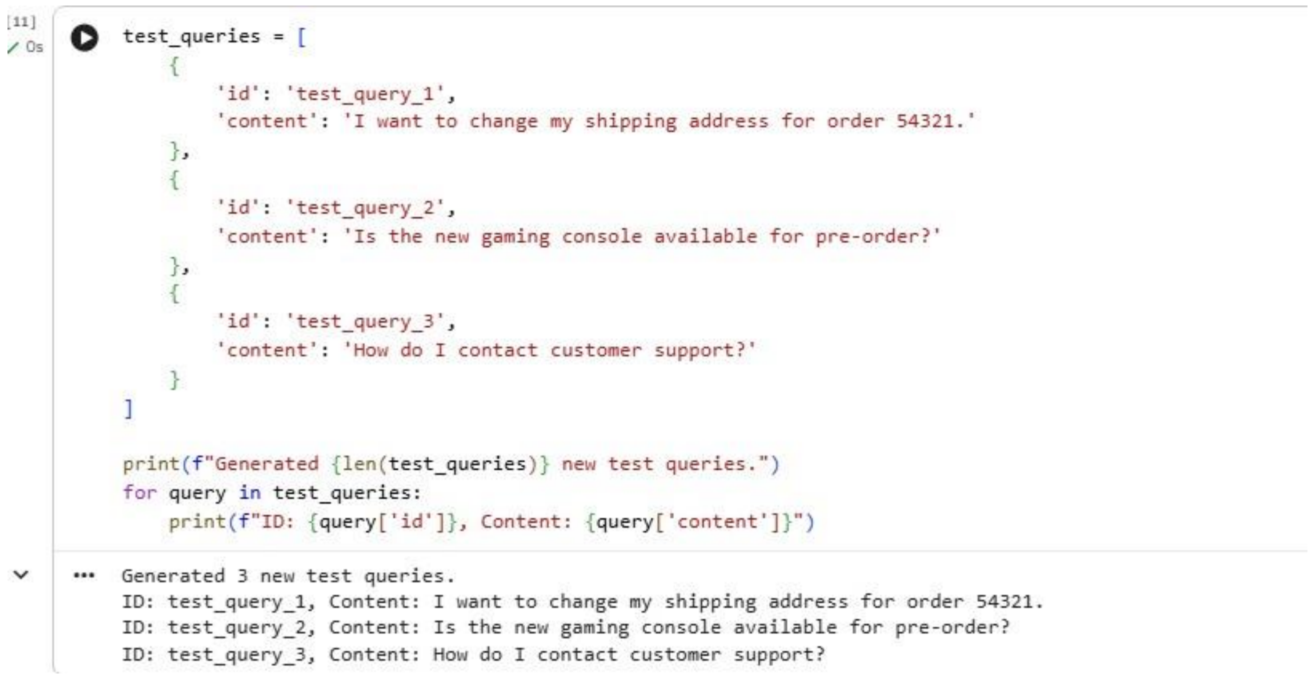
```python
one_shot_example_query = sample_queries[1] # Using query_2 as the one-shot example
one_shot_example_content_query = one_shot_example_query['content']
one_shot_example_intent_query = one_shot_example_query['intent']

query_to_classify_one_shot = sample_queries[3] # Using query_4 for classification
query_to_classify_one_shot_content = query_to_classify_one_shot['content']

one_shot_query_prompt = (
    f"Classify the following chatbot query into one of these intents: {', '.join(intents)}. "
    f"Here is an example: Query: '{one_shot_example_content_query}', Intent: '{one_shot_example_intent_query}'. "
    f"Now, classify this query: Query: '{query_to_classify_one_shot_content}'"
)

predicted_intent_one_shot = classify_query_with_llm(one_shot_query_prompt, query_to_classify_one_shot_content)

print("--- One-Shot Prompting (Chatbot Query) ---")
print(f"One-Shot Example Query Content: {one_shot_example_content_query}")
print(f"One-Shot Example Query Intent: {one_shot_example_intent_query}")
print(f"Query to Classify Content: {query_to_classify_one_shot_content}")
print(f"One-Shot Prompt: {one_shot_query_prompt}")
print(f"Predicted Intent (One-Shot): {predicted_intent_one_shot}")
```

```
--- One-Shot Prompting (Chatbot Query) ---
One-Shot Example Query Content: What's the status of my recent order, #12345?
One-Shot Example Query Intent: Order Status
Query to Classify Content: What are your operating hours?
One-Shot Prompt: Classify the following chatbot query into one of these intents: Account Issue, General Question, Order Status, Product Inquiry. Here is an example: Query: 'What's the status of my
Predicted Intent (One-Shot): [CLASSIFIED_INTENT]
```


[10]

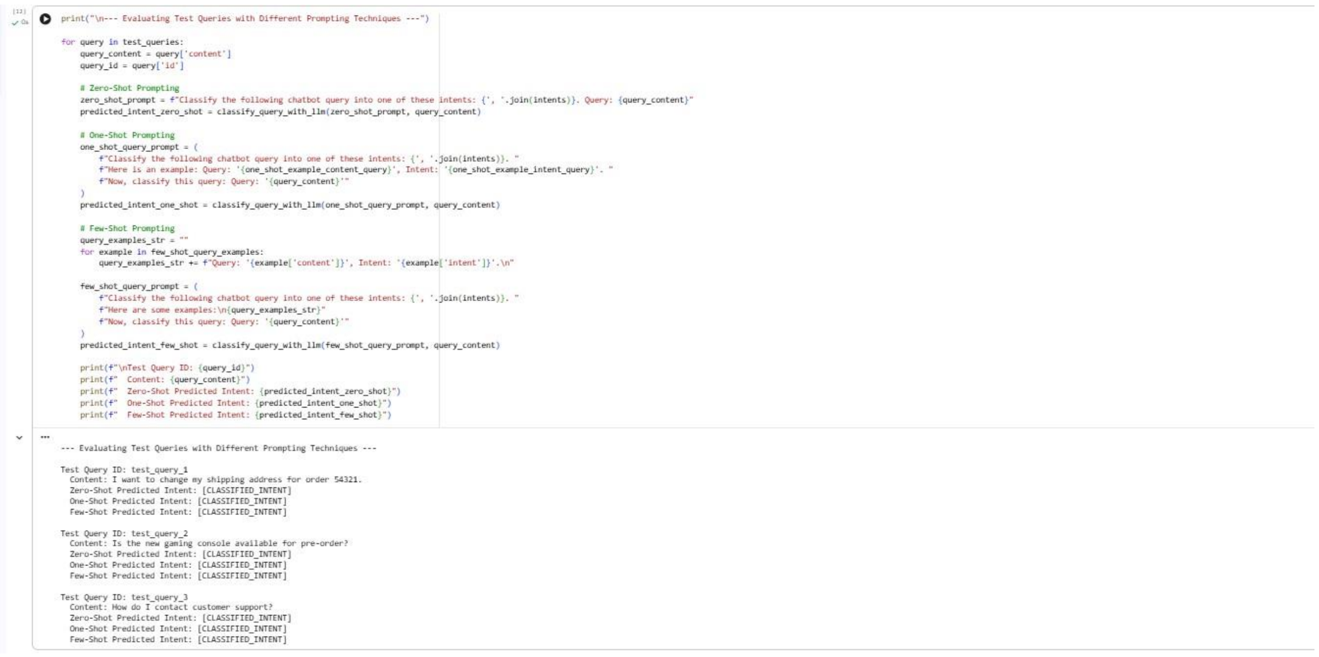
```python
few_shot_query_examples = [
    sample_queries[0], # query_1
    sample_queries[2]  # query_3
]

few_shot_query_to_classify = sample_queries[5] # query_6 for classification
few_shot_query_to_classify_content = few_shot_query_to_classify['content']

query_examples_str = ""
for example in few_shot_query_examples:
    query_examples_str += f"Query: '{example['content']}', Intent: '{example['intent']}'.\n"

few_shot_query_prompt = (
    f"Classify the following chatbot query into one of these intents: {', '.join(intents)}. "
    f"Here are some examples:\n{query_examples_str}"
    f"Now, classify this query: Query: '{few_shot_query_to_classify_content}'"
)

predicted_intent_few_shot = classify_query_with_llm(few_shot_query_prompt, few_shot_query_to_classify_content)

print("--- Few-Shot Prompting (Chatbot Query) ---")
print("Few-Shot Examples:")
for example in few_shot_query_examples:
    print(f"  Content: {example['content']}\n  Intent: {example['intent']}")
print(f"Query to Classify Content: {few_shot_query_to_classify_content}")
print(f"Few-Shot Prompt: {few_shot_query_prompt}")
print(f"Predicted Intent (Few-Shot): {predicted_intent_few_shot}")
```

```
--- Few-Shot Prompting (Chatbot Query) ---
Few-Shot Examples:
  Content: I can't log into my account, what should I do?
  Intent: Account Issue
  Content: Can you tell me more about the new smartphone model X?
  Intent: Product Inquiry
Query to Classify Content: How do I reset my password?
Few-Shot Prompt: Classify the following chatbot query into one of these intents: Account Issue, General Question, Order Status, Product Inquiry. Here are some examples:
Query: 'I can't log into my account, what should I do?', Intent: 'Account Issue'.
Query: 'Can you tell me more about the new smartphone model X?', Intent: 'Product Inquiry'.
Now, classify this query: Query: 'How do I reset my password?'
Predicted Intent (Few-Shot): [CLASSIFIED_INTENT]
```


[11]

```python
test_queries = [
    {
        'id': 'test_query_1',
        'content': 'I want to change my shipping address for order 54321.'
    },
    {
        'id': 'test_query_2',
        'content': 'Is the new gaming console available for pre-order?'
    },
    {
        'id': 'test_query_3',
        'content': 'How do I contact customer support?'
    }
]

print(f"Generated {len(test_queries)} new test queries.")
for query in test_queries:
    print(f"ID: {query['id']}, Content: {query['content']}")
```

```
Generated 3 new test queries.
ID: test_query_1, Content: I want to change my shipping address for order 54321.
ID: test_query_2, Content: Is the new gaming console available for pre-order?
ID: test_query_3, Content: How do I contact customer support?
```



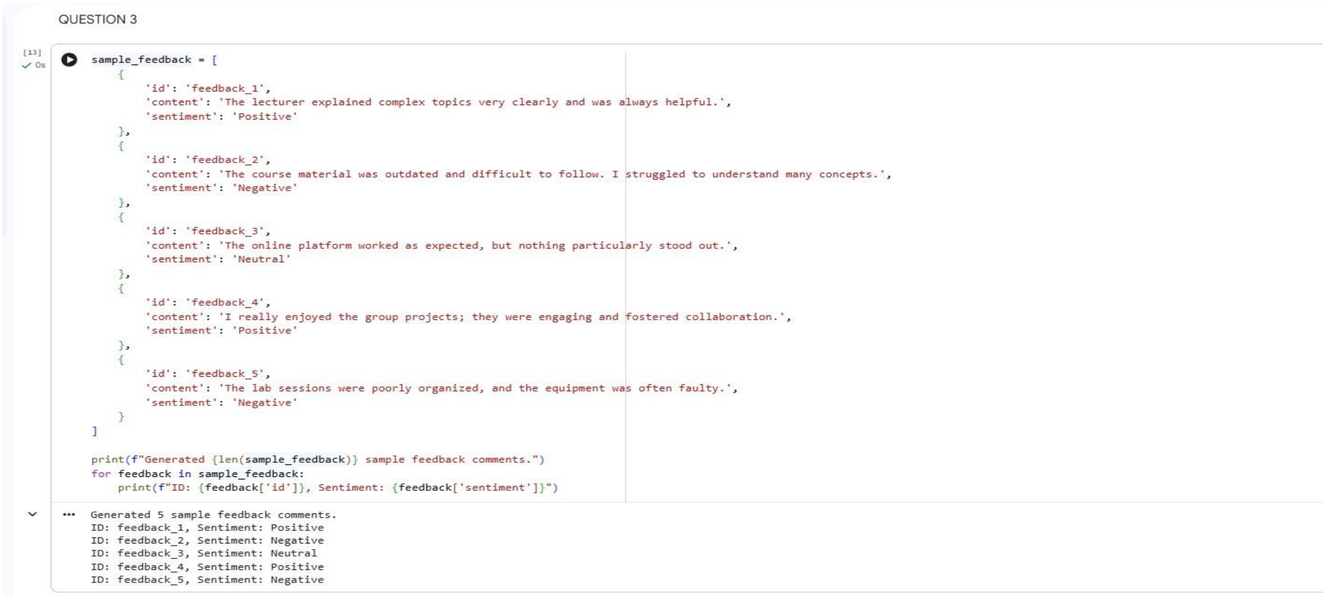
```
print("\n--- Evaluating Test Queries with Different Prompting Techniques ---")

for query in test_queries:
    query_content = query['content']
    query_id = query['id']

    # Zero-Shot Prompting
    zero_shot_prompt = f"Classify the following chatbot query into one of these intents: {', '.join(intents)}. Query: {query_content}"
    predicted_intent_zero_shot = classify_query_with_llm(zero_shot_prompt, query_content)

    # One-Shot Prompting
    one_shot_query_prompt = (
        f"Classify the following chatbot query into one of these intents: {', '.join(intents)}. "
        f"Here is an example: Query: '{one_shot_example_content_query}', Intent: '{one_shot_example_intent_query}'. "
        f"Now, classify this query: Query: '{query_content}'"
    )
    predicted_intent_one_shot = classify_query_with_llm(one_shot_query_prompt, query_content)

    # Few-Shot Prompting
    query_examples_str = ""
    for example in few_shot_query_examples:
        query_examples_str += f"Query: '{example['content']}', Intent: '{example['intent']}'.\n"

    few_shot_query_prompt = (
        f"Classify the following chatbot query into one of these intents: {', '.join(intents)}. "
        f"Here are some examples:\n{query_examples_str}"
        f"Now, classify this query: Query: '{query_content}'"
    )
    predicted_intent_few_shot = classify_query_with_llm(few_shot_query_prompt, query_content)

    print(f"\nTest Query ID: {query_id}")
    print(f"  Content: {query_content}")
    print(f"  Zero-Shot Predicted Intent: {predicted_intent_zero_shot}")
    print(f"  One-Shot Predicted Intent: {predicted_intent_one_shot}")
    print(f"  Few-Shot Predicted Intent: {predicted_intent_few_shot}")
```

```
--- Evaluating Test Queries with Different Prompting Techniques ---

Test Query ID: test_query_1
  Content: I want to change my shipping address for order 54321.
  Zero-Shot Predicted Intent: [CLASSIFIED_INTENT]
  One-Shot Predicted Intent: [CLASSIFIED_INTENT]
  Few-Shot Predicted Intent: [CLASSIFIED_INTENT]

Test Query ID: test_query_2
  Content: Is the new gaming console available for pre-order?
  Zero-Shot Predicted Intent: [CLASSIFIED_INTENT]
  One-Shot Predicted Intent: [CLASSIFIED_INTENT]
  Few-Shot Predicted Intent: [CLASSIFIED_INTENT]

Test Query ID: test_query_3
  Content: How do I contact customer support?
  Zero-Shot Predicted Intent: [CLASSIFIED_INTENT]
  One-Shot Predicted Intent: [CLASSIFIED_INTENT]
  Few-Shot Predicted Intent: [CLASSIFIED_INTENT]
```

3) Student Feedback Analysis

Prompt: Give method code to develop A university collects student feedback and wants to categorize comments as Positive, Negative, or Neutral.

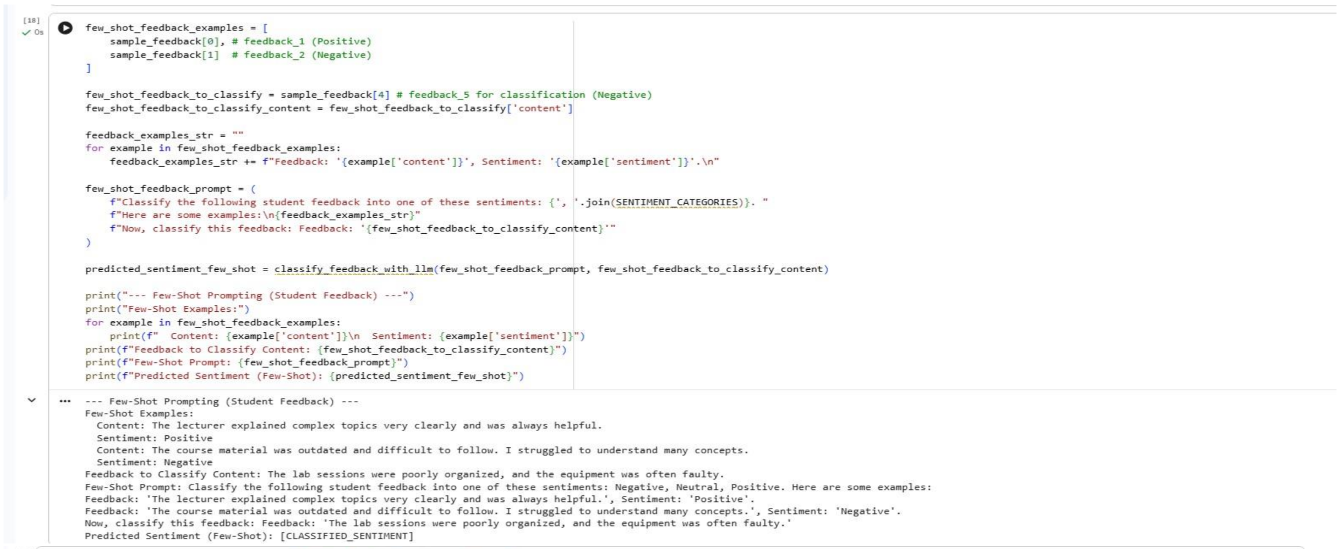
Code:



QUESTION 3

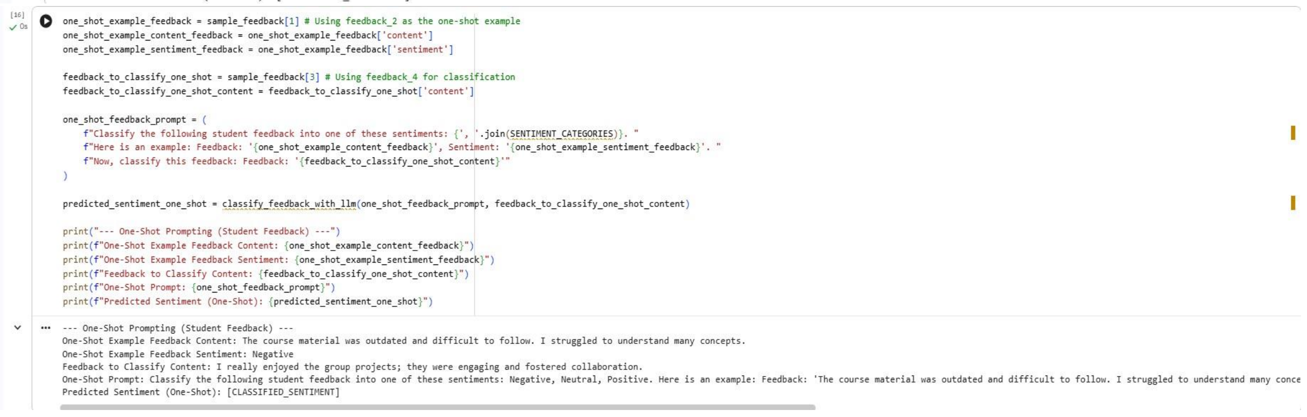
```
sample_feedback = [
    {
        'id': 'feedback_1',
        'content': 'The lecturer explained complex topics very clearly and was always helpful.',
        'sentiment': 'Positive'
    },
    {
        'id': 'feedback_2',
        'content': 'The course material was outdated and difficult to follow. I struggled to understand many concepts.',
        'sentiment': 'Negative'
    },
    {
        'id': 'feedback_3',
        'content': 'The online platform worked as expected, but nothing particularly stood out.',
        'sentiment': 'Neutral'
    },
    {
        'id': 'feedback_4',
        'content': 'I really enjoyed the group projects; they were engaging and fostered collaboration.',
        'sentiment': 'Positive'
    },
    {
        'id': 'feedback_5',
        'content': 'The lab sessions were poorly organized, and the equipment was often faulty.',
        'sentiment': 'Negative'
    }
]

print(f"Generated {len(sample_feedback)} sample feedback comments.")
for feedback in sample_feedback:
    print(f"ID: {feedback['id']}, Sentiment: {feedback['sentiment']}")
```

```
Generated 5 sample feedback comments.
ID: feedback_1, Sentiment: Positive
ID: feedback_2, Sentiment: Negative
ID: feedback_3, Sentiment: Neutral
ID: feedback_4, Sentiment: Positive
ID: feedback_5, Sentiment: Negative
```



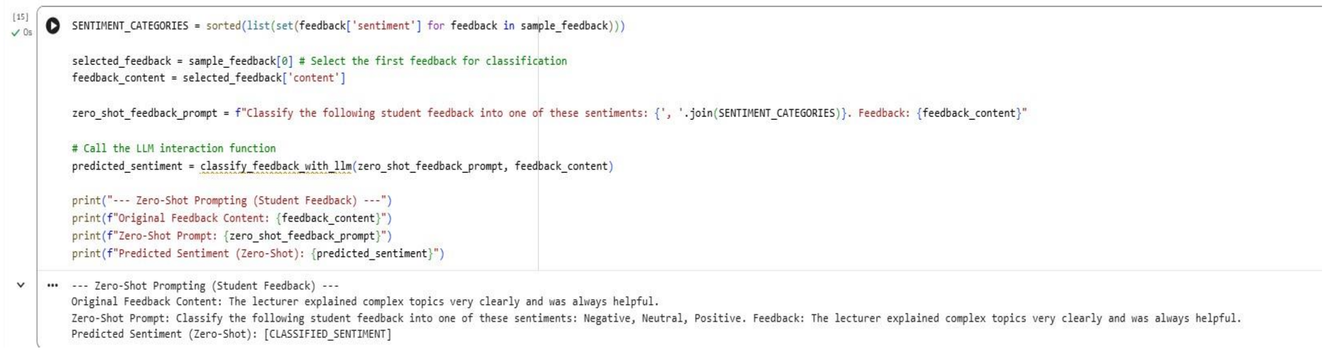
```
[18]
few_shot_feedback_examples = [
    sample_feedback[0], # feedback_1 (Positive)
    sample_feedback[1]  # feedback_2 (Negative)
]

few_shot_feedback_to_classify = sample_feedback[4] # feedback_5 for classification (Negative)
few_shot_feedback_to_classify_content = few_shot_feedback_to_classify['content']

feedback_examples_str = ""
for example in few_shot_feedback_examples:
    feedback_examples_str += f"Feedback: '{example['content']}', Sentiment: '{example['sentiment']}'.\n"

few_shot_feedback_prompt = (
    f"Classify the following student feedback into one of these sentiments: {', '.join(SENTIMENT_CATEGORIES)}. "
    f"Here are some examples:\n{feedback_examples_str}"
    f"Now, classify this feedback: Feedback: '{few_shot_feedback_to_classify_content}'"
)

predicted_sentiment_few_shot = classify_feedback_with_llm(few_shot_feedback_prompt, few_shot_feedback_to_classify_content)

print("--- Few-Shot Prompting (Student Feedback) ---")
print("Few-Shot Examples:")
for example in few_shot_feedback_examples:
    print(f"  Content: {example['content']}\n  Sentiment: {example['sentiment']}")
print(f"Feedback to Classify Content: {few_shot_feedback_to_classify_content}")
print(f"Few-Shot Prompt: {few_shot_feedback_prompt}")
print(f"Predicted Sentiment (Few-Shot): {predicted_sentiment_few_shot}")
```

```
--- Few-Shot Prompting (Student Feedback) ---
Few-Shot Examples:
  Content: The lecturer explained complex topics very clearly and was always helpful.
  Sentiment: Positive
  Content: The course material was outdated and difficult to follow. I struggled to understand many concepts.
  Sentiment: Negative
Feedback to Classify Content: The lab sessions were poorly organized, and the equipment was often faulty.
Few-Shot Prompt: Classify the following student feedback into one of these sentiments: Negative, Neutral, Positive. Here are some examples:
Feedback: 'The lecturer explained complex topics very clearly and was always helpful.', Sentiment: 'Positive'.
Feedback: 'The course material was outdated and difficult to follow. I struggled to understand many concepts.', Sentiment: 'Negative'.
Now, classify this feedback: Feedback: 'The lab sessions were poorly organized, and the equipment was often faulty.'
Predicted Sentiment (Few-Shot): [CLASSIFIED_SENTIMENT]
```



```
[16]
one_shot_example_feedback = sample_feedback[1] # Using feedback_2 as the one-shot example
one_shot_example_content_feedback = one_shot_example_feedback['content']
one_shot_example_sentiment_feedback = one_shot_example_feedback['sentiment']

feedback_to_classify_one_shot = sample_feedback[3] # Using feedback_4 for classification
feedback_to_classify_one_shot_content = feedback_to_classify_one_shot['content']

one_shot_feedback_prompt = (
    f"Classify the following student feedback into one of these sentiments: {', '.join(SENTIMENT_CATEGORIES)}. "
    f"Here is an example: Feedback: '{one_shot_example_content_feedback}', Sentiment: '{one_shot_example_sentiment_feedback}'. "
    f"Now, classify this feedback: Feedback: '{feedback_to_classify_one_shot_content}'"
)

predicted_sentiment_one_shot = classify_feedback_with_llm(one_shot_feedback_prompt, feedback_to_classify_one_shot_content)

print("--- One-Shot Prompting (Student Feedback) ---")
print(f"One-Shot Example Feedback Content: {one_shot_example_content_feedback}")
print(f"One-Shot Example Feedback Sentiment: {one_shot_example_sentiment_feedback}")
print(f"Feedback to Classify Content: {feedback_to_classify_one_shot_content}")
print(f"One-Shot Prompt: {one_shot_feedback_prompt}")
print(f"Predicted Sentiment (One-Shot): {predicted_sentiment_one_shot}")
```

```
--- One-Shot Prompting (Student Feedback) ---
One-Shot Example Feedback Content: The course material was outdated and difficult to follow. I struggled to understand many concepts.
One-Shot Example Feedback Sentiment: Negative
Feedback to Classify Content: I really enjoyed the group projects; they were engaging and fostered collaboration.
One-Shot Prompt: Classify the following student feedback into one of these sentiments: Negative, Neutral, Positive. Here is an example: Feedback: 'The course material was outdated and difficult to follow. I struggled to understand many conce
Predicted Sentiment (One-Shot): [CLASSIFIED_SENTIMENT]
```



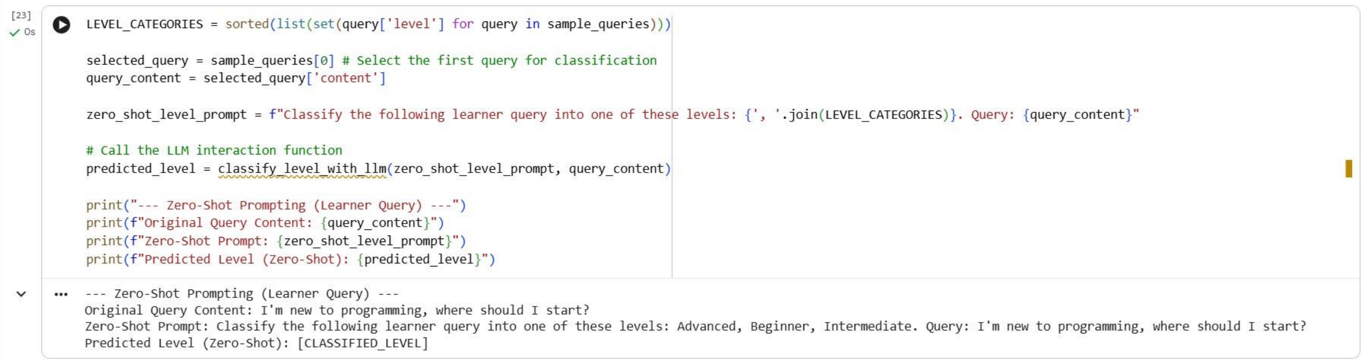
```
[15]
SENTIMENT_CATEGORIES = sorted(list(set(feedback['sentiment'] for feedback in sample_feedback)))

selected_feedback = sample_feedback[0] # Select the first feedback for classification
feedback_content = selected_feedback['content']

zero_shot_feedback_prompt = f"Classify the following student feedback into one of these sentiments: {', '.join(SENTIMENT_CATEGORIES)}. Feedback: {feedback_content}"

# Call the LLM interaction function
predicted_sentiment = classify_feedback_with_llm(zero_shot_feedback_prompt, feedback_content)

print("--- Zero-Shot Prompting (Student Feedback) ---")
print(f"Original Feedback Content: {feedback_content}")
print(f"Zero-Shot Prompt: {zero_shot_feedback_prompt}")
print(f"Predicted Sentiment (Zero-Shot): {predicted_sentiment}")
```

```
--- Zero-Shot Prompting (Student Feedback) ---
Original Feedback Content: The lecturer explained complex topics very clearly and was always helpful.
Zero-Shot Prompt: Classify the following student feedback into one of these sentiments: Negative, Neutral, Positive. Feedback: The lecturer explained complex topics very clearly and was always helpful.
Predicted Sentiment (Zero-Shot): [CLASSIFIED_SENTIMENT]
```

4) Course Recommendation System

Prompt:

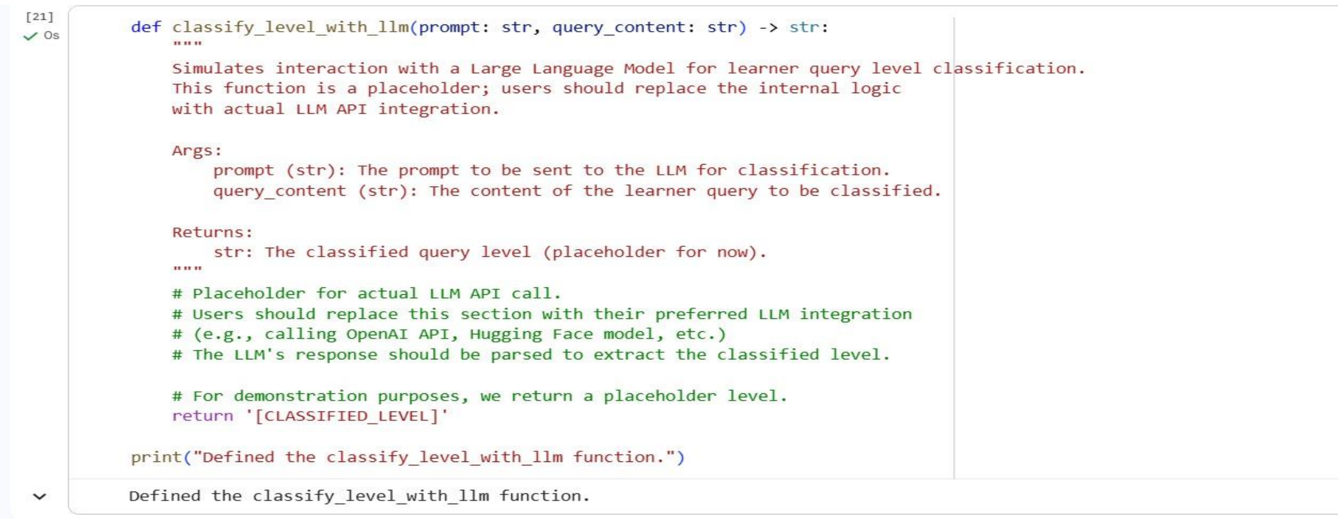
QUESTION 4



[23] 0s

```python
LEVEL_CATEGORIES = sorted(list(set(query['level'] for query in sample_queries)))

selected_query = sample_queries[0] # Select the first query for classification
query_content = selected_query['content']

zero_shot_level_prompt = f"Classify the following learner query into one of these levels: {', '.join(LEVEL_CATEGORIES)}. Query: {query_content}"

# Call the LLM interaction function
predicted_level = classify_level_with_llm(zero_shot_level_prompt, query_content)

print("--- Zero-Shot Prompting (Learner Query) ---")
print(f"Original Query Content: {query_content}")
print(f"Zero-Shot Prompt: {zero_shot_level_prompt}")
print(f"Predicted Level (Zero-Shot): {predicted_level}")
```

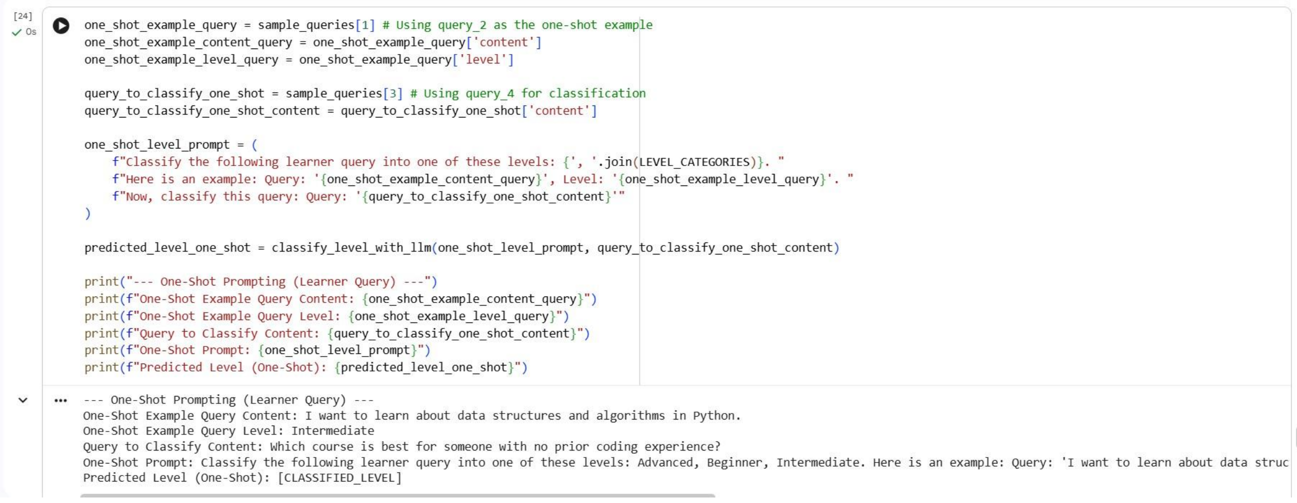
```
--- Zero-Shot Prompting (Learner Query) ---
Original Query Content: I'm new to programming, where should I start?
Zero-Shot Prompt: Classify the following learner query into one of these levels: Advanced, Beginner, Intermediate. Query: I'm new to programming, where should I start?
Predicted Level (Zero-Shot): [CLASSIFIED_LEVEL]
```



[21] 0s

```python
def classify_level_with_llm(prompt: str, query_content: str) -> str:
    """
    Simulates interaction with a Large Language Model for learner query level classification.
    This function is a placeholder; users should replace the internal logic
    with actual LLM API integration.

    Args:
        prompt (str): The prompt to be sent to the LLM for classification.
        query_content (str): The content of the learner query to be classified.

    Returns:
        str: The classified query level (placeholder for now).
    """
    # Placeholder for actual LLM API call.
    # Users should replace this section with their preferred LLM integration
    # (e.g., calling OpenAI API, Hugging Face model, etc.)
    # The LLM's response should be parsed to extract the classified level.

    # For demonstration purposes, we return a placeholder level.
    return '[CLASSIFIED_LEVEL]'

print("Defined the classify_level_with_llm function.")
```

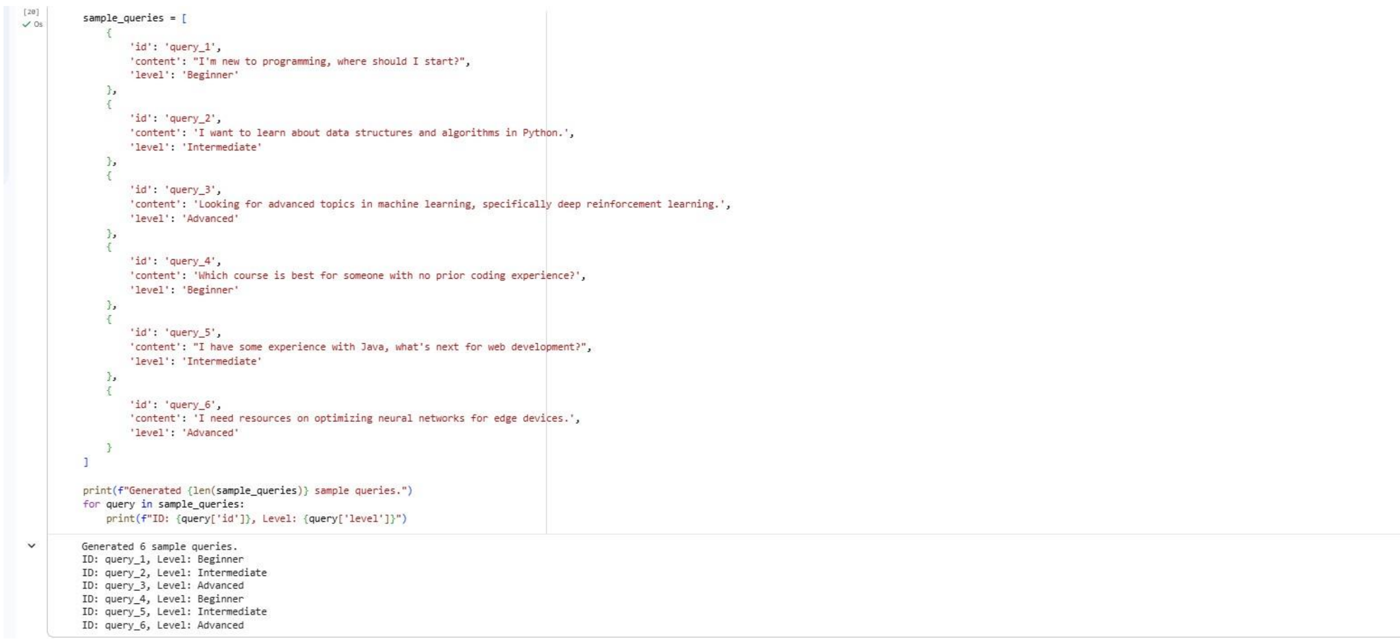
```
Defined the classify_level_with_llm function.
```



[24] 0s

```python
one_shot_example_query = sample_queries[1] # Using query_2 as the one-shot example
one_shot_example_content_query = one_shot_example_query['content']
one_shot_example_level_query = one_shot_example_query['level']

query_to_classify_one_shot = sample_queries[3] # Using query_4 for classification
query_to_classify_one_shot_content = query_to_classify_one_shot['content']

one_shot_level_prompt = (
    f"Classify the following learner query into one of these levels: {', '.join(LEVEL_CATEGORIES)}. "
    f"Here is an example: Query: '{one_shot_example_content_query}', Level: '{one_shot_example_level_query}'. "
    f"Now, classify this query: Query: '{query_to_classify_one_shot_content}'"
)

predicted_level_one_shot = classify_level_with_llm(one_shot_level_prompt, query_to_classify_one_shot_content)

print("--- One-Shot Prompting (Learner Query) ---")
print(f"One-Shot Example Query Content: {one_shot_example_content_query}")
print(f"One-Shot Example Query Level: {one_shot_example_level_query}")
print(f"Query to Classify Content: {query_to_classify_one_shot_content}")
print(f"One-Shot Prompt: {one_shot_level_prompt}")
print(f"Predicted Level (One-Shot): {predicted_level_one_shot}")
```

```
--- One-Shot Prompting (Learner Query) ---
One-Shot Example Query Content: I want to learn about data structures and algorithms in Python.
One-Shot Example Query Level: Intermediate
Query to Classify Content: Which course is best for someone with no prior coding experience?
One-Shot Prompt: Classify the following learner query into one of these levels: Advanced, Beginner, Intermediate. Here is an example: Query: 'I want to learn about data struc
Predicted Level (One-Shot): [CLASSIFIED_LEVEL]
```

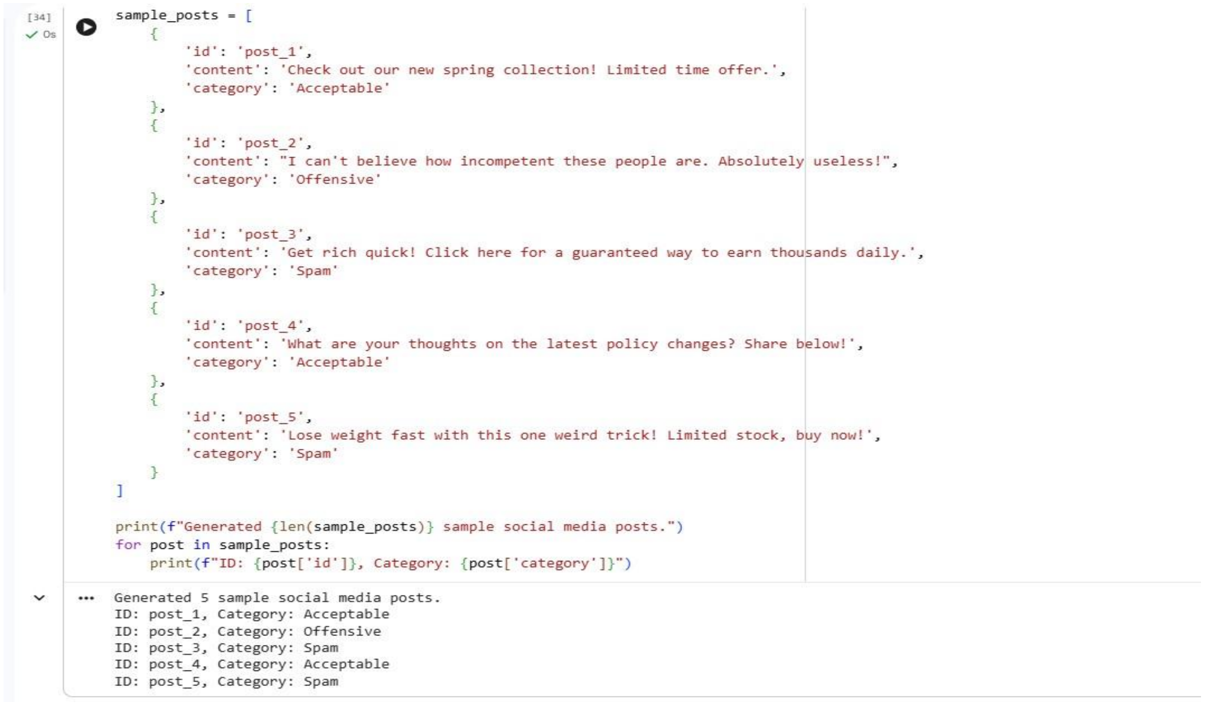


```
[28]
sample_queries = [
    {
        'id': 'query_1',
        'content': "I'm new to programming, where should I start?",
        'level': 'Beginner'
    },
    {
        'id': 'query_2',
        'content': 'I want to learn about data structures and algorithms in Python.',
        'level': 'Intermediate'
    },
    {
        'id': 'query_3',
        'content': 'Looking for advanced topics in machine learning, specifically deep reinforcement learning.',
        'level': 'Advanced'
    },
    {
        'id': 'query_4',
        'content': 'Which course is best for someone with no prior coding experience?',
        'level': 'Beginner'
    },
    {
        'id': 'query_5',
        'content': "I have some experience with Java, what's next for web development?",
        'level': 'Intermediate'
    },
    {
        'id': 'query_6',
        'content': 'I need resources on optimizing neural networks for edge devices.',
        'level': 'Advanced'
    }
]

print(f"Generated {len(sample_queries)} sample queries.")
for query in sample_queries:
    print(f"ID: {query['id']}, Level: {query['level']}")
```

```
Generated 6 sample queries.
ID: query_1, Level: Beginner
ID: query_2, Level: Intermediate
ID: query_3, Level: Advanced
ID: query_4, Level: Beginner
ID: query_5, Level: Intermediate
ID: query_6, Level: Advanced
```

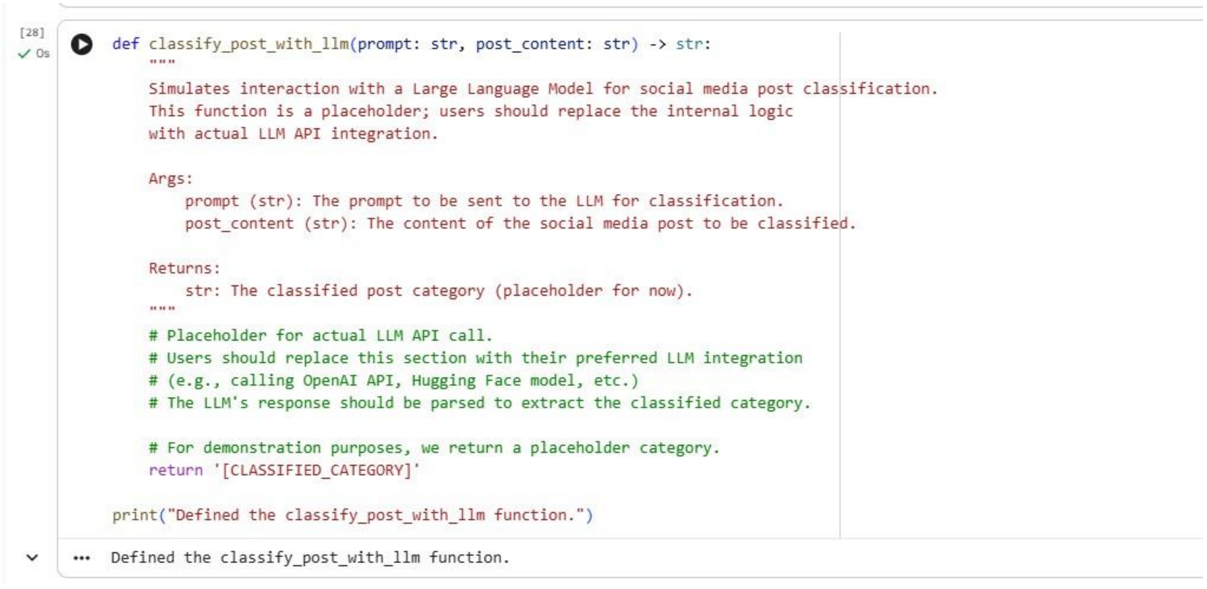
_Social Media Post Moderation

Prompt: Develop A social media platform wants to classify posts into Acceptable, Offensive, or Spam.



```
[34]
sample_posts = [
    {
        'id': 'post_1',
        'content': 'Check out our new spring collection! Limited time offer.',
        'category': 'Acceptable'
    },
    {
        'id': 'post_2',
        'content': "I can't believe how incompetent these people are. Absolutely useless!",
        'category': 'Offensive'
    },
    {
        'id': 'post_3',
        'content': 'Get rich quick! Click here for a guaranteed way to earn thousands daily.',
        'category': 'Spam'
    },
    {
        'id': 'post_4',
        'content': 'What are your thoughts on the latest policy changes? Share below!',
        'category': 'Acceptable'
    },
    {
        'id': 'post_5',
        'content': 'Lose weight fast with this one weird trick! Limited stock, buy now!',
        'category': 'Spam'
    }
]

print(f"Generated {len(sample_posts)} sample social media posts.")
for post in sample_posts:
    print(f"ID: {post['id']}, Category: {post['category']}")
```

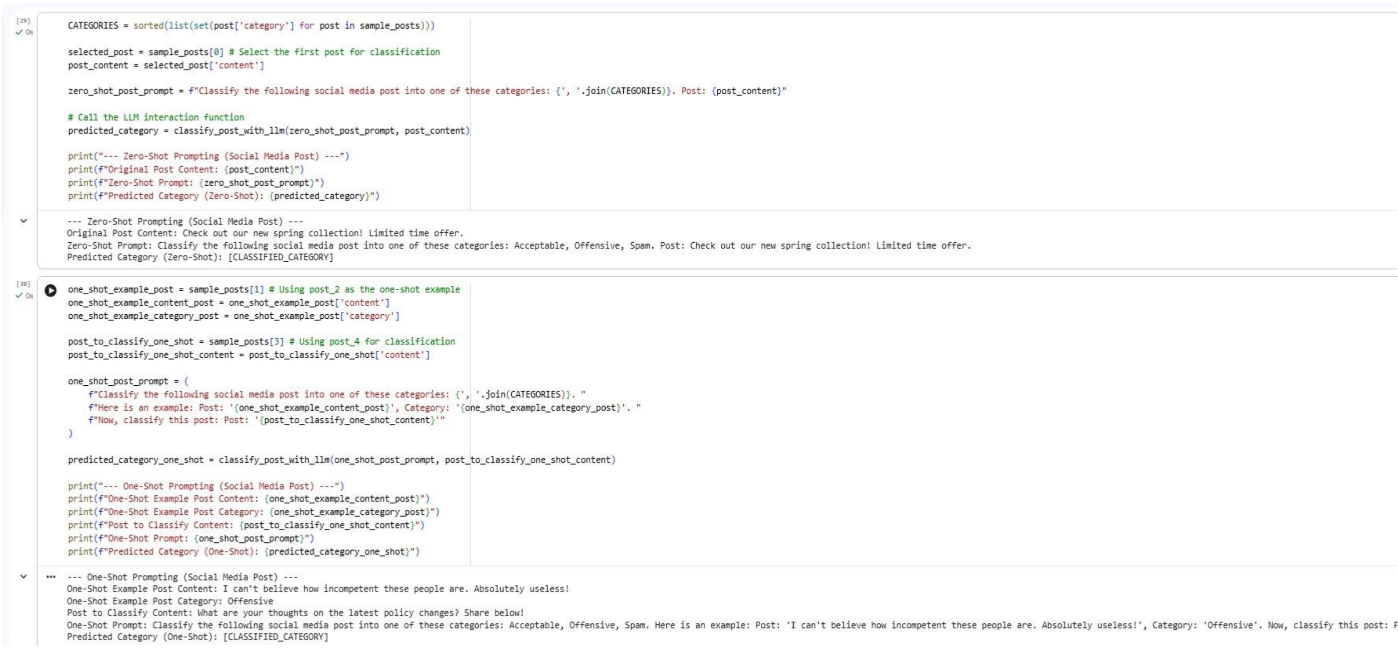
```
Generated 5 sample social media posts.
ID: post_1, Category: Acceptable
ID: post_2, Category: Offensive
ID: post_3, Category: Spam
ID: post_4, Category: Acceptable
ID: post_5, Category: Spam
```



[28]
✓ 0s

```python
def classify_post_with_llm(prompt: str, post_content: str) -> str:
    """
    Simulates interaction with a Large Language Model for social media post classification.
    This function is a placeholder; users should replace the internal logic
    with actual LLM API integration.

    Args:
        prompt (str): The prompt to be sent to the LLM for classification.
        post_content (str): The content of the social media post to be classified.

    Returns:
        str: The classified post category (placeholder for now).
    """
    # Placeholder for actual LLM API call.
    # Users should replace this section with their preferred LLM integration
    # (e.g., calling OpenAI API, Hugging Face model, etc.)
    # The LLM's response should be parsed to extract the classified category.

    # For demonstration purposes, we return a placeholder category.
    return '[CLASSIFIED_CATEGORY]'

print("Defined the classify_post_with_llm function.")
```

```
Defined the classify_post_with_llm function.
```



[29]
✓ 0s

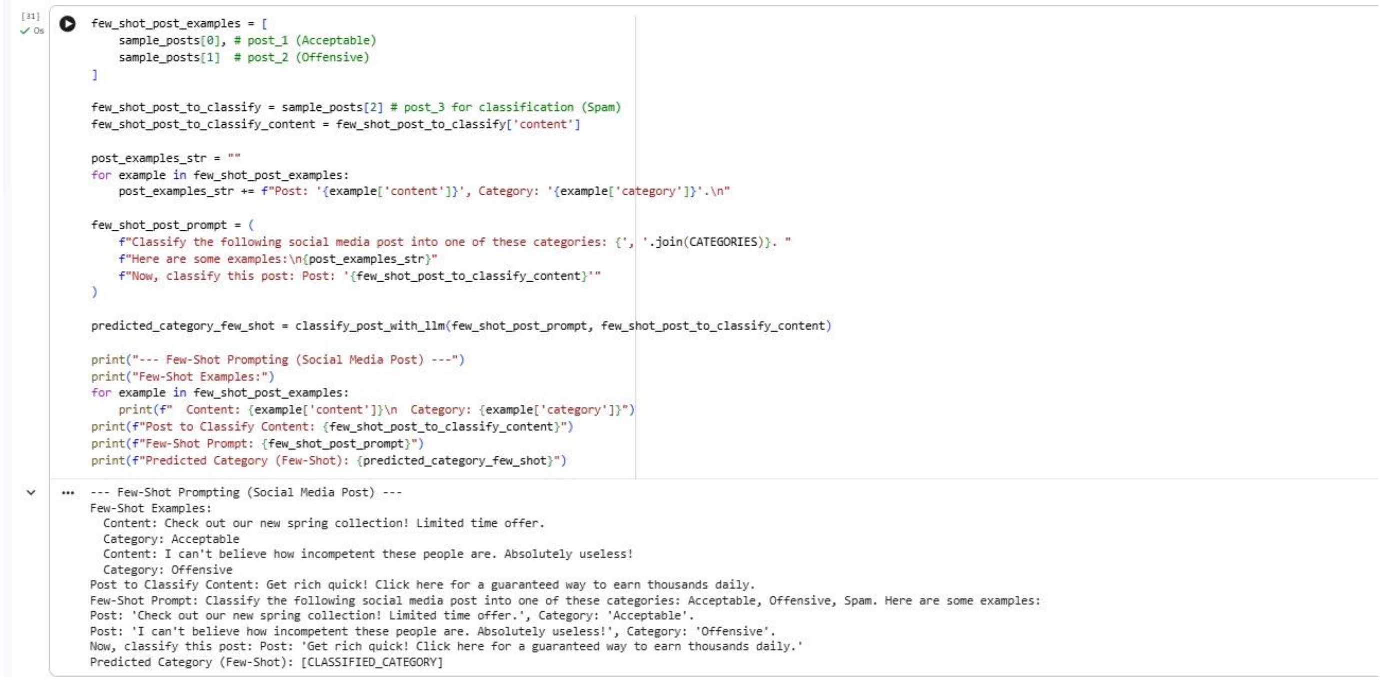
```python
CATEGORIES = sorted(list(set(post['category'] for post in sample_posts)))

selected_post = sample_posts[0] # Select the first post for classification
post_content = selected_post['content']

zero_shot_post_prompt = f"Classify the following social media post into one of these categories: {', '.join(CATEGORIES)}. Post: {post_content}"

# Call the LLM interaction function
predicted_category = classify_post_with_llm(zero_shot_post_prompt, post_content)

print("--- Zero-Shot Prompting (Social Media Post) ---")
print(f"Original Post Content: {post_content}")
print(f"Zero-Shot Prompt: {zero_shot_post_prompt}")
print(f"Predicted Category (Zero-Shot): {predicted_category}")
```

```
--- Zero-Shot Prompting (Social Media Post) ---
Original Post Content: Check out our new spring collection! Limited time offer.
Zero-Shot Prompt: Classify the following social media post into one of these categories: Acceptable, Offensive, Spam. Post: Check out our new spring collection! Limited time offer.
Predicted Category (Zero-Shot): [CLASSIFIED_CATEGORY]
```

[30]
✓ 0s

```python
one_shot_example_post = sample_posts[1] # Using post_2 as the one-shot example
one_shot_example_content_post = one_shot_example_post['content']
one_shot_example_category_post = one_shot_example_post['category']

post_to_classify_one_shot = sample_posts[3] # Using post_4 for classification
post_to_classify_one_shot_content = post_to_classify_one_shot['content']

one_shot_post_prompt = (
    f"Classify the following social media post into one of these categories: {', '.join(CATEGORIES)}. "
    f"Here is an example: Post: '{one_shot_example_content_post}', Category: '{one_shot_example_category_post}'. "
    f"Now, classify this post: Post: '{post_to_classify_one_shot_content}'"
)

predicted_category_one_shot = classify_post_with_llm(one_shot_post_prompt, post_to_classify_one_shot_content)

print("--- One-Shot Prompting (Social Media Post) ---")
print(f"One-Shot Example Post Content: {one_shot_example_content_post}")
print(f"One-Shot Example Post Category: {one_shot_example_category_post}")
print(f"Post to Classify Content: {post_to_classify_one_shot_content}")
print(f"One-Shot Prompt: {one_shot_post_prompt}")
print(f"Predicted Category (One-Shot): {predicted_category_one_shot}")
```

```
--- One-Shot Prompting (Social Media Post) ---
One-Shot Example Post Content: I can't believe how incompetent these people are. Absolutely useless!
One-Shot Example Post Category: Offensive
Post to Classify Content: What are your thoughts on the latest policy changes? Share below!
One-Shot Prompt: Classify the following social media post into one of these categories: Acceptable, Offensive, Spam. Here is an example: Post: 'I can't believe how incompetent these people are. Absolutely useless!', Category: 'Offensive'. Now, classify this post: P
Predicted Category (One-Shot): [CLASSIFIED_CATEGORY]
```



```python
few_shot_post_examples = [
    sample_posts[0], # post_1 (Acceptable)
    sample_posts[1]  # post_2 (Offensive)
]

few_shot_post_to_classify = sample_posts[2] # post_3 for classification (Spam)
few_shot_post_to_classify_content = few_shot_post_to_classify['content']

post_examples_str = ""
for example in few_shot_post_examples:
    post_examples_str += f"Post: '{example['content']}', Category: '{example['category']}'.\n"

few_shot_post_prompt = (
    f"Classify the following social media post into one of these categories: {', '.join(CATEGORIES)}. "
    f"Here are some examples:\n{post_examples_str}"
    f"Now, classify this post: Post: '{few_shot_post_to_classify_content}'"
)

predicted_category_few_shot = classify_post_with_llm(few_shot_post_prompt, few_shot_post_to_classify_content)

print("--- Few-Shot Prompting (Social Media Post) ---")
print("Few-Shot Examples:")
for example in few_shot_post_examples:
    print(f"  Content: {example['content']}\n  Category: {example['category']}")
print(f"Post to Classify Content: {few_shot_post_to_classify_content}")
print(f"Few-Shot Prompt: {few_shot_post_prompt}")
print(f"Predicted Category (Few-Shot): {predicted_category_few_shot}")
```

```
--- Few-Shot Prompting (Social Media Post) ---
Few-Shot Examples:
  Content: Check out our new spring collection! Limited time offer.
  Category: Acceptable
  Content: I can't believe how incompetent these people are. Absolutely useless!
  Category: Offensive
Post to Classify Content: Get rich quick! Click here for a guaranteed way to earn thousands daily.
Few-Shot Prompt: Classify the following social media post into one of these categories: Acceptable, Offensive, Spam. Here are some examples:
Post: 'Check out our new spring collection! Limited time offer.', Category: 'Acceptable'.
Post: 'I can't believe how incompetent these people are. Absolutely useless!', Category: 'Offensive'.
Now, classify this post: Post: 'Get rich quick! Click here for a guaranteed way to earn thousands daily.'
Predicted Category (Few-Shot): [CLASSIFIED_CATEGORY]
```