

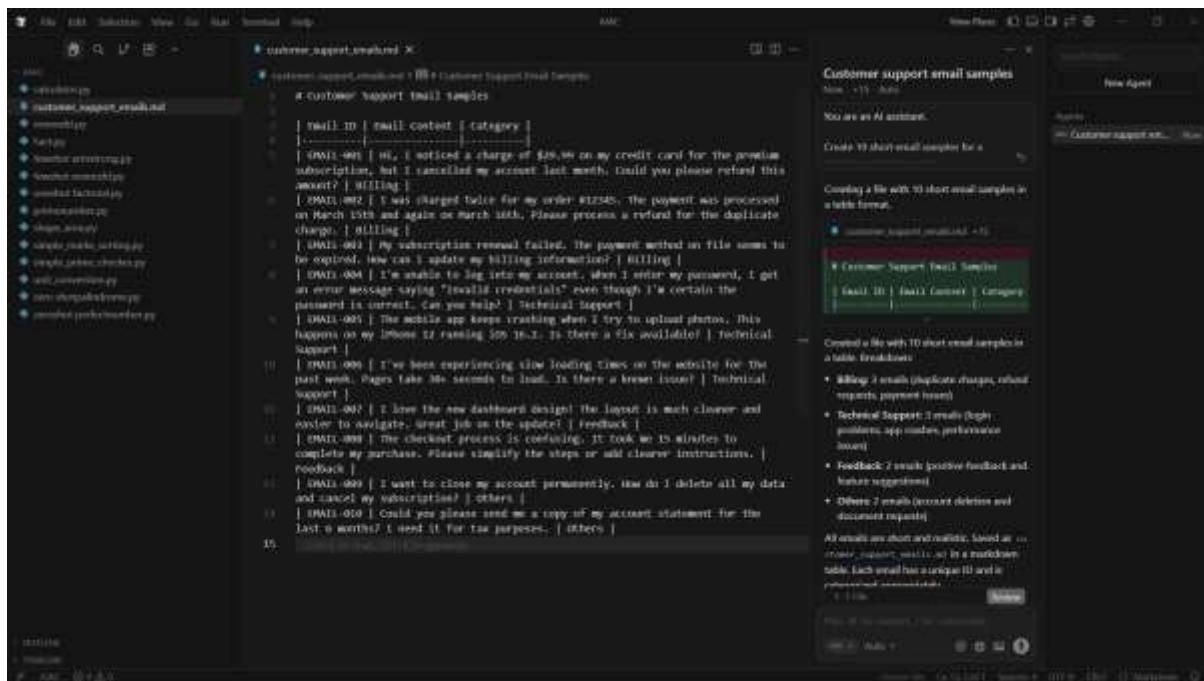
School of Computer Science and Artificial Intelligence

Lab Assignment -4.5

Program	:B. Tech (CSE)
Specialization	:AIML
Course Title	: AI ASSISTED CODING
Course Code	: 23CS002PC304
Semester	: VI
Academic Session	: 2025-2026
Name of Student	: K. Shailaja
Enrollment No.	: 2303A52155
Batch No.	: 34

a. Prepare Sample Data:

- Create or collect 10 short email samples, each belonging to one of the 4 categories.



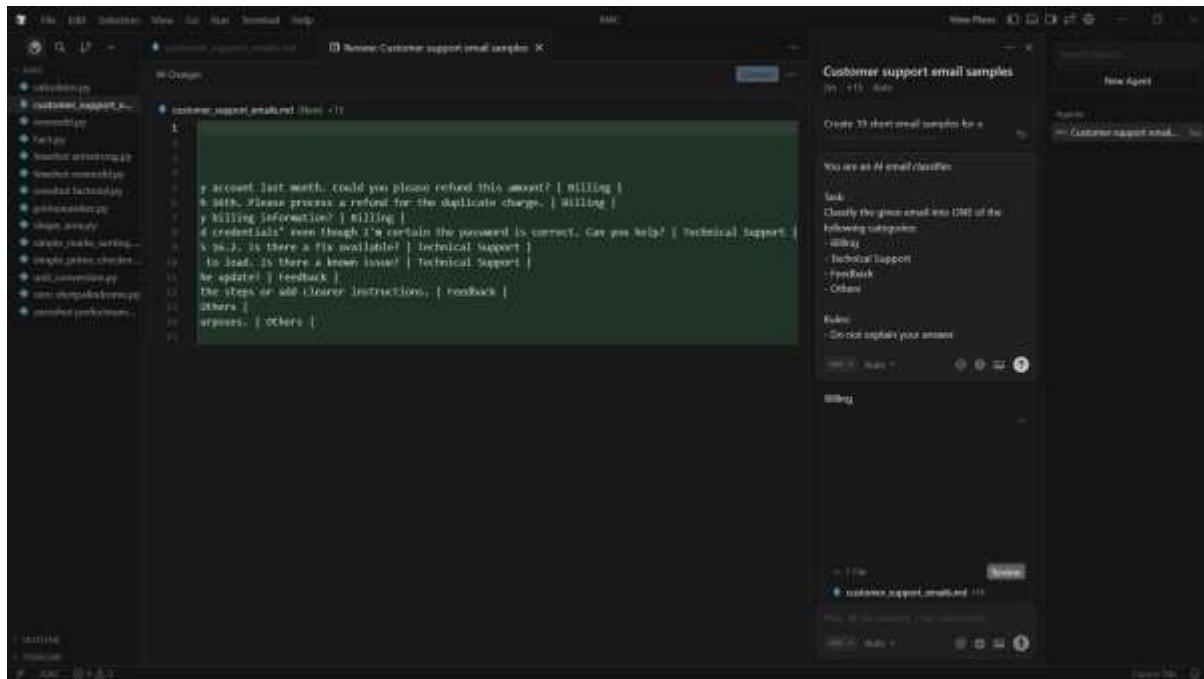
b. Zero-shot Prompting:

- Design a prompt that asks the LLM to classify a single email without providing any examples.

- Example prompt:

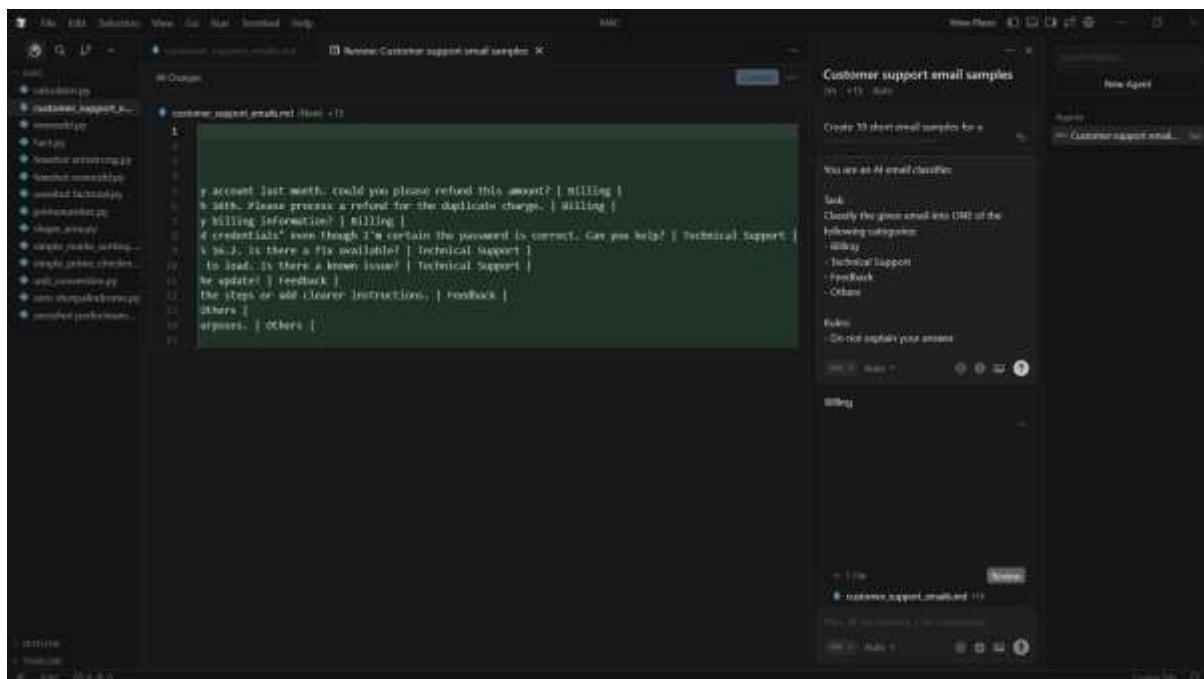
"Classify the following email into one of the following categories:

Billing, Technical Support, Feedback, Others. Email: 'I have not received my invoice for last month.'



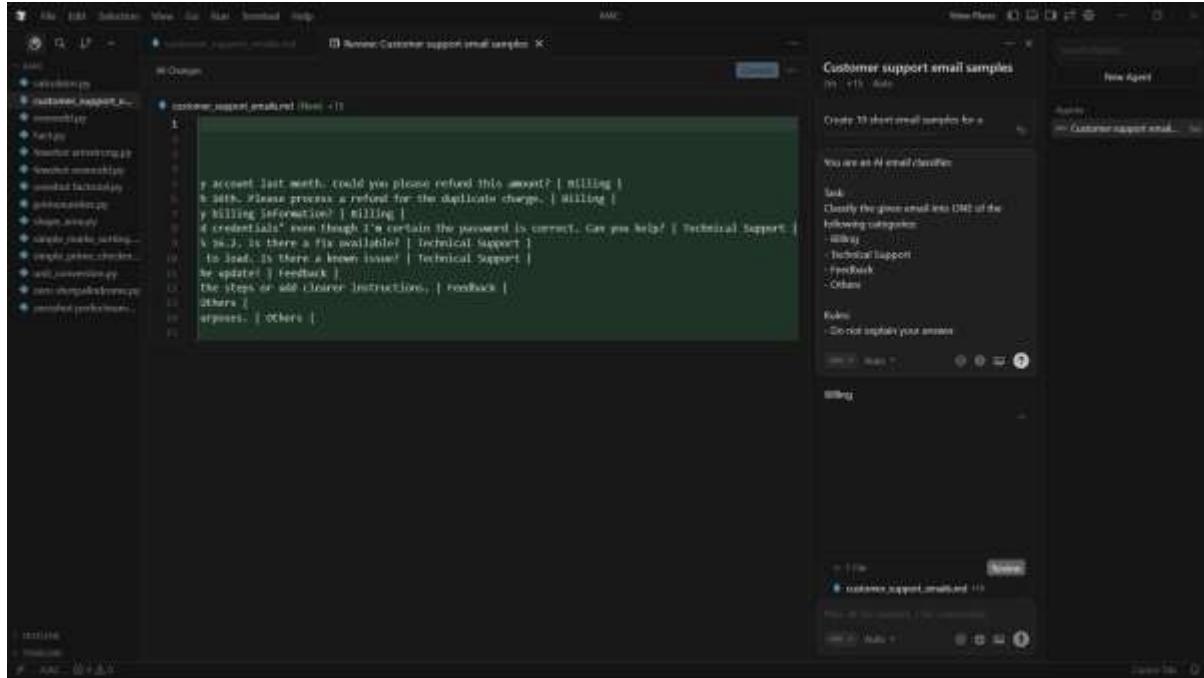
c. One-shot Prompting:

- Add one labeled example before asking the model to classify a new email.



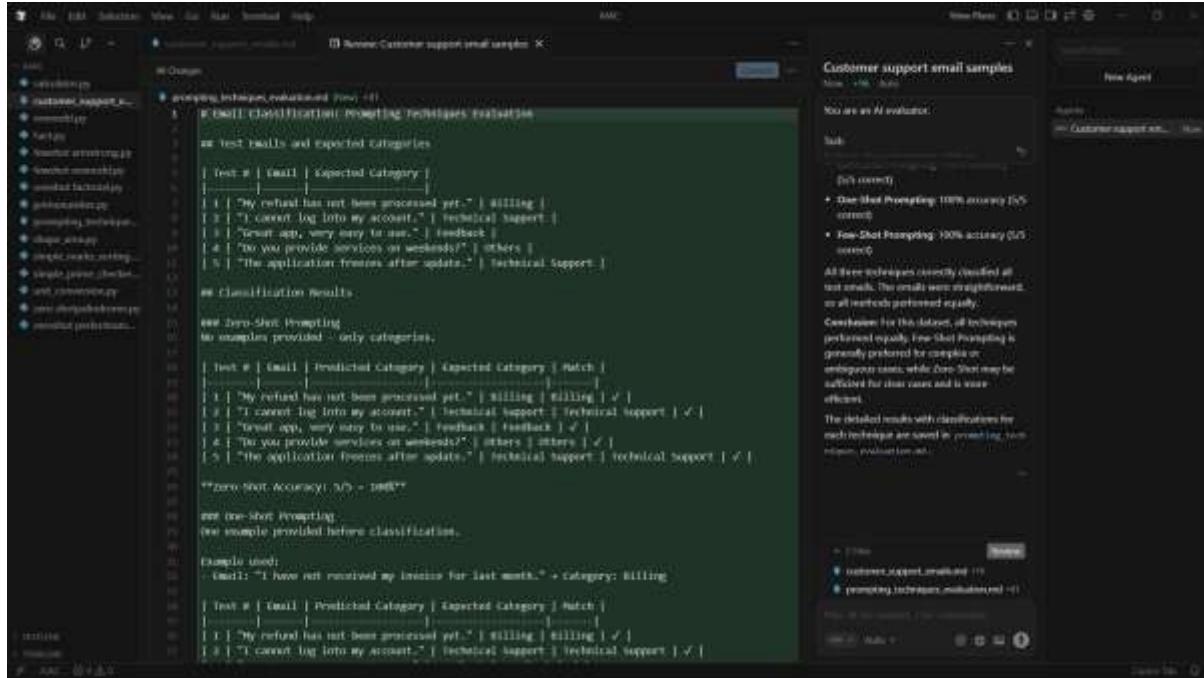
d. Few-shot Prompting:

- Use 3–5 labeled examples in your prompt before asking the model to classify a new email.



e. Evaluation:

- Run all three techniques on the same set of 5 test emails.
- Compare and document the accuracy and clarity of responses.



```

# Examples used:
# Email: "I have not received my invoice for last month." -> Category: Billing
# Email: "The application crashes when I open it." -> Category: Technical Support
# Email: "I really like the new update of your app." -> Category: Feedback
# Email: "What are your office working hours?" -> Category: Others

| test # | email | predicted category | expected category | match |
| --- | --- | --- | --- | --- |
| 1 | "My refund has not been processed yet." | Billing | Billing | ✓ |
| 2 | "I cannot log into my account." | Technical Support | Technical Support | ✓ |
| 3 | "Great app, very easy to use." | Feedback | Feedback | ✓ |
| 4 | "Do you provide services on weekends?" | Others | Others | ✓ |
| 5 | "The application freezes after update." | Technical Support | Technical Support | ✓ |

**Zero-Shot Accuracy: 5/5 = 100%**  

**Summary Table:**


| Technique           | Correct Predictions | Total Emails | Accuracy |
|---------------------|---------------------|--------------|----------|
| Zero-Shot Prompting | 5                   | 5            | 100%     |
| Few-Shot Prompting  | 5                   | 5            | 100%     |
| One-Shot Prompting  | 5                   | 5            | 100%     |


**Conclusion:**
All three prompting techniques achieved perfect accuracy (100%) on this test set. The test emails can be categorized correctly by all methods. Zero-Shot is the most efficient, followed by Few-Shot and One-Shot. All methods perform equally well on this dataset. However, few-shot prompting may be more efficient for complex or ambiguous cases, while zero-shot may be sufficient for clear cases and is more efficient.

```

You are an AI evaluator.
Task:
 Is it correct?
• One-Shot Prompting: 100% accuracy (5/5 correct)
• Few-Shot Prompting: 100% accuracy (5/5 correct)
All three techniques correctly classified all test emails. The results were straightforward, so all methods performed equally.
Conclusion: For this dataset, all techniques performed equally. Few-Shot Prompting is generally preferred for complex or ambiguous cases, while Zero-Shot may be sufficient for clear cases and is more efficient.
The detailed results with classifications for each technique are saved in [prompting_small_evaluation.csv](#).

2. Travel Query Classification

Scenario:

A travel assistant must classify queries into Flight Booking, Hotel Booking, Cancellation, or General Travel Info.

Tasks:

- a. Prepare labeled travel queries.
- b. Apply Zero-shot prompting.
- c. Apply One-shot prompting.
- d. Apply Few-shot prompting.
- e. Compare response consistency.

```
nbconvert --to html travel_query_annotation.ipynb
```

Travel query annotation

Recent notebooks

1 travel_query_annotation.ipynb (0:100 words)

query": "I want to book a flight from Delhi to Mumbai.",
category": "Flight booking"

query": "Show me available flights for tomorrow.",
category": "Flight booking"

query": "I need a hotel near the airport in Bangalore.",
category": "Hotel booking"

query": "Book a 3-star hotel in Chennai for two nights.",
category": "Hotel booking"

query": "Cancelled my flight scheduled for next Monday.",
category": "Cancellation"

query": "I want to cancel my hotel reservation.",
category": "Cancellation"

query": "What documents are required for international travel?",
category": "General travel info"

query": "What is the baggage allowance for domestic flights?",
category": "General travel info"

Travel assistant query labeling

Create labeled travel queries for a travel assistant system

A. travel_query_annotation.ipynb - CSV file

Both files include:

- 14 categories: Flight booking, Hotel booking, Cancellation, General travel info
- 8 queries with their corresponding categories

The dataset can be used for:

- Training classification models
- Building intent recognition systems
- Building a travel assistant chatbot
- Data analysis and evaluation

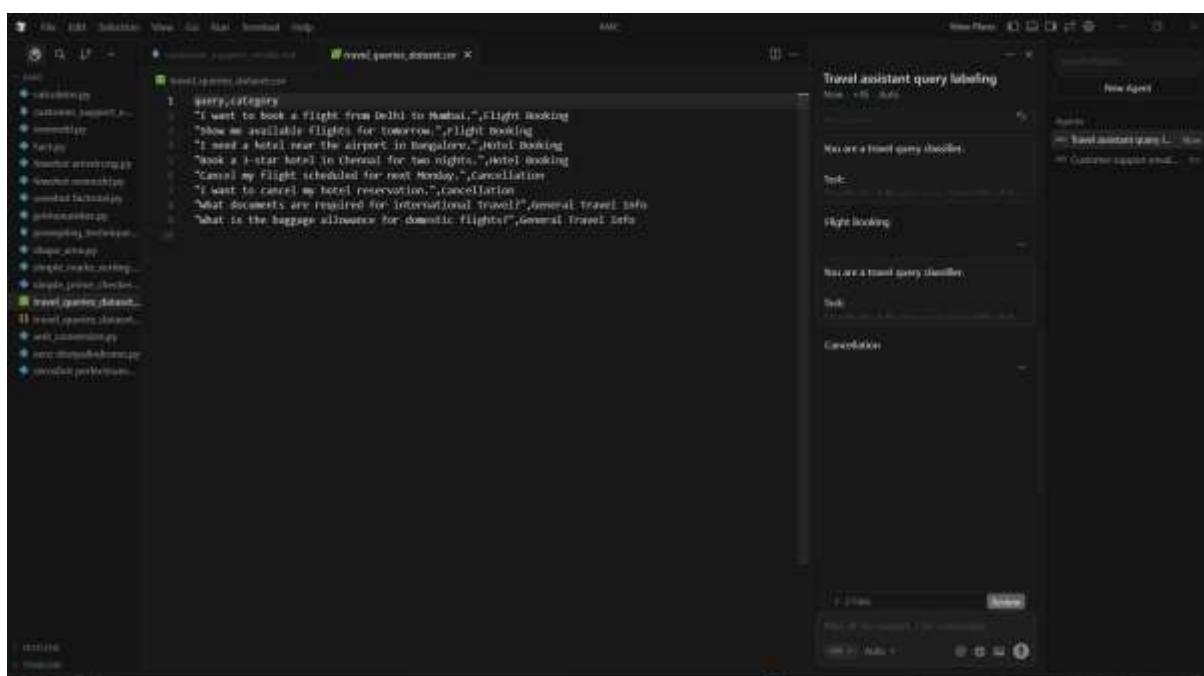
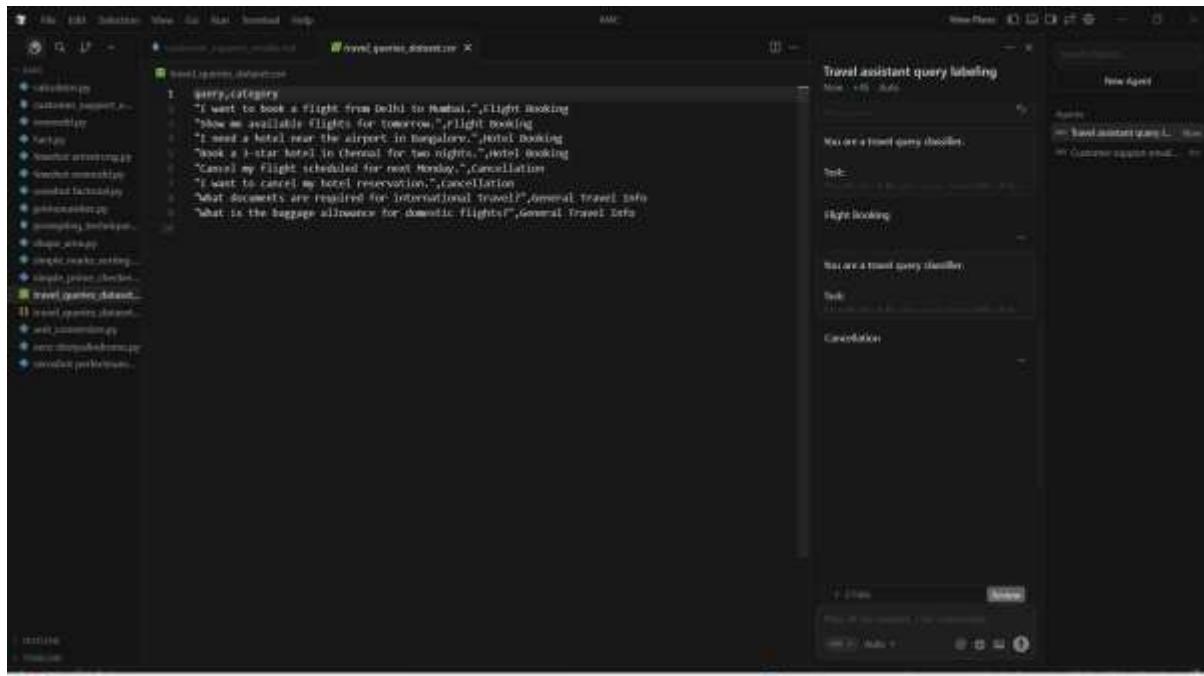
Should I add more queries or create a Python script to load and process this dataset?

travel_query_annotation.ipynb

The screenshot shows the Oracle Database SQL Developer interface. The left sidebar displays a tree view of database objects, including schema, tables, and views. A node for 'travel_query_classifier' is expanded, showing several entries under 'query category'. The main pane contains a SQL query:

```
1 /* query category
   2 "I want to book a flight from Delhi to Mumbai.", "Flight booking
   3 "Show me available flights for tomorrow", "Flight booking
   4 "I need a hotel near the airport in Bangalore.", "Hotel booking
   5 "Book a 3-star hotel in Bengaluru for two nights.", "Hotel booking
   6 "Cancel my flight scheduled for next Monday", "Cancellation
   7 "I want to cancel my hotel reservation", "Cancellation
   8 "What documents are required for international travel?", "General travel info
   9 "What is the baggage allowance for domestic flights?", "General travel info
```

To the right, a 'Travel assistant query labeling' panel is open, showing a list of labeled queries and a section for creating new labels. The bottom right corner shows the standard Oracle Database SQL Developer status bar.



The screenshot shows a Jupyter Notebook interface with two tabs: "Travel Query Classifier Evaluation" and "Untitled". The "Travel Query Classifier Evaluation" tab contains Python code for classifying travel queries into categories like flight booking, hotel booking, cancellation, and general travel info. It includes functions for generating test queries and zero-shot classification. A sidebar on the right displays a "Travel assistant query labeling" interface with a text input field, a progress bar, and a "Run command python" button.

```
class TravelQueryClassifierEvaluation:
    def __init__(self):
        self.test_queries = [
            "Book a flight from Ahmedabad to New Delhi",
            "Find a budget hotel in New Delhi",
            "Cancel my hotel reservation",
            "What documents are required for international travel?",
            "I want to cancel my flight."
        ]
        self.expected_categories = [
            "Flight Booking",
            "Hotel Booking",
            "Cancellation",
            "General Travel Info",
            "General"
        ]
        self.categories = [
            "Flight Booking",
            "Hotel Booking",
            "Cancellation",
            "General Travel Info"
        ]

    def zero_shot_classify(query):
        ...
        zero_shot_prompts_only_instructions_no_examples
        prompt = f'''You are a travel query classifier.
        TASK: Classify the following query into one of the categories:
        {query}'''

    def eval_results():
        for result in [zero_shot_results, one_shot_results, few_shot_results]:
            for i, result in enumerate(result['results']):
                print()
                print(">" * 100)
                print(result['category'])
                print(">" * 100)

                accuracy = result['accuracy']
                zero_shot_results['technique'], zero_shot_results['accuracy'],
                one_shot_results['technique'], one_shot_results['accuracy'],
                few_shot_results['technique'], few_shot_results['accuracy']

            best_technique = max(accuracy, inv_accuracy.get())
            best_accuracy = accuracy[best_technique]

            print(f'Default classification technique: {best_technique}')
            print(f'Accuracy: {best_accuracy:.4f}%')
            print()

        if few_shot_results['accuracy'] > one_shot_results['accuracy'] > zero_shot_results['accuracy']:
            print('Observation: Few-shot prompting demonstrates the highest consistency, followed by One-shot, and then Zero-shot. Providing examples helps the model better understand the classification.')
        elif one_shot_results['accuracy'] > zero_shot_results['accuracy']:
            print('Observation: One-shot prompting shows better performance than zero-shot, indicating the prompt even with a single example can improve classification consistency.')
        else:
            print('Observation: Results vary across techniques. Additional examples may help improve consistency.')

        print(">" * 100)
        print(results)
```

This screenshot shows the continuation of the Jupyter Notebook code for the Travel Query Classifier Evaluation. The code defines a function to evaluate results, comparing zero-shot, one-shot, and few-shot classification accuracies. It prints the best technique and accuracy for each shot type and provides an observation about the consistency of results across different prompting techniques.

```
def eval_results():
    for result in [zero_shot_results, one_shot_results, few_shot_results]:
        for i, result in enumerate(result['results']):
            print()
            print(">" * 100)
            print(result['category'])
            print(">" * 100)

            accuracy = result['accuracy']
            zero_shot_results['technique'], zero_shot_results['accuracy'],
            one_shot_results['technique'], one_shot_results['accuracy'],
            few_shot_results['technique'], few_shot_results['accuracy']

        best_technique = max(accuracy, inv_accuracy.get())
        best_accuracy = accuracy[best_technique]

        print(f'Default classification technique: {best_technique}')
        print(f'Accuracy: {best_accuracy:.4f}%')
        print()

    if few_shot_results['accuracy'] > one_shot_results['accuracy'] > zero_shot_results['accuracy']:
        print('Observation: Few-shot prompting demonstrates the highest consistency, followed by One-shot, and then Zero-shot. Providing examples helps the model better understand the classification.')
    elif one_shot_results['accuracy'] > zero_shot_results['accuracy']:
        print('Observation: One-shot prompting shows better performance than zero-shot, indicating the prompt even with a single example can improve classification consistency.')
    else:
        print('Observation: Results vary across techniques. Additional examples may help improve consistency.')

    print(">" * 100)
    print(results)
```

3. Programming Question Type Identification

Scenario:

A coding help chatbot must classify queries into Syntax Error, Logic

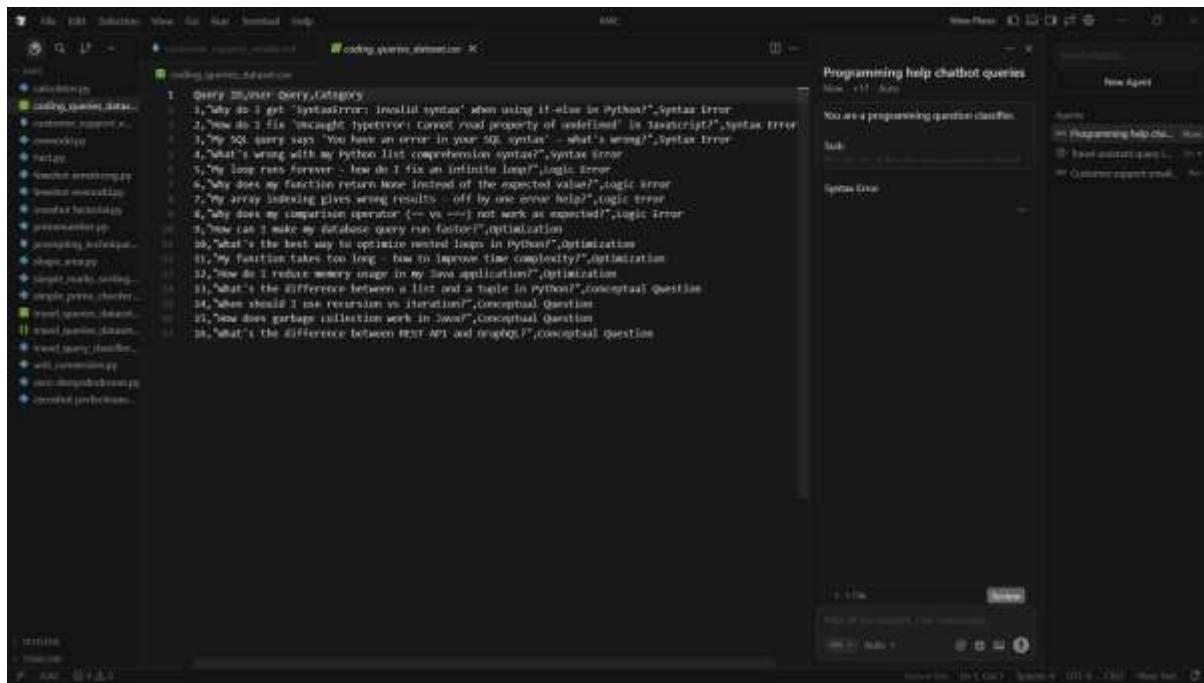
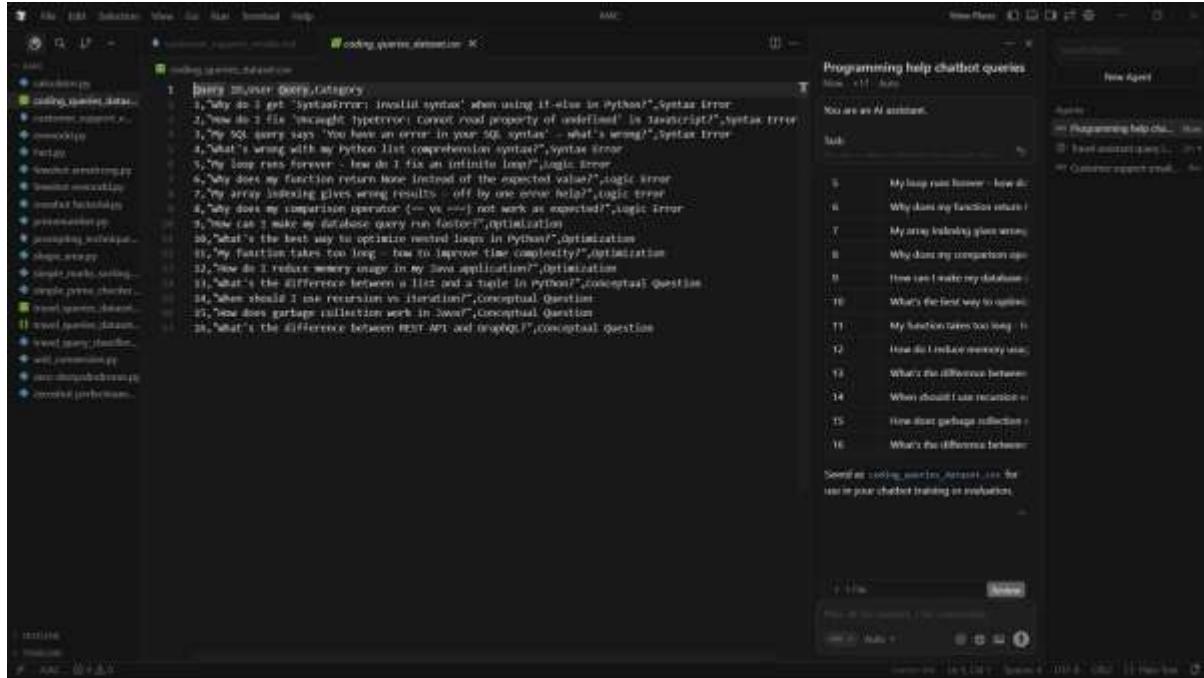
Error, Optimization, or Conceptual Question.

Tasks:

a. Prepare coding-related user queries.

b. Perform Zero-shot classification.

- c. Perform One-shot classification.
 - d. Perform Few-shot classification.
 - e. Analyze improvements in technical accuracy.



Programming help chatbot queries

New Agent

You are a programming question classifier.

Task

Topic

Logic Error

Optimization

1. QHRY 2019-01-01T00:00:00Z 00000000-0000-0000-0000-000000000000

1. "Why do I get 'SyntaxError: invalid syntax' when using if-else in python?",Syntax Error

2. "How do I fix 'Uncaught TypeError: Cannot read property of undefined' in javascript?",Syntax Error

3. "My SQL query says 'You have an error in your SQL syntax - what's wrong?' Syntax Error

4. "What's wrong with my Python list comprehension syntax?",Syntax Error

5. "My loop runs forever - how do I fix an infinite loop?",Logic Error

6. "My function returns None instead of the expected value?",Logic Error

7. "My array filtering gives wrong results - off by one error help?",Logic Error

8. "Why does my comparison operator (<= vs. ==) not work as expected?",Logic Error

9. "How can I make my database query run faster?",Optimization

10. "My function takes too long - how to improve time complexity?",Optimization

11. "What's the difference between a list and a tuple in python?",Conceptual Question

12. "When should I use recursion vs. iteration?",Conceptual Question

13. "How does garbage collection work in Java?",Conceptual Question

14. "What's the difference between REST API and GraphQL?",Conceptual Question

Topic

Optimization

1. QHRY 2019-01-01T00:00:00Z 00000000-0000-0000-0000-000000000000

1. "Why do I get 'SyntaxError: invalid syntax' when using if-else in python?",Syntax Error

2. "How do I fix 'Uncaught TypeError: Cannot read property of undefined' in javascript?",Syntax Error

3. "My SQL query says 'You have an error in your SQL syntax - what's wrong?' Syntax Error

4. "What's wrong with my Python list comprehension syntax?",Syntax Error

5. "My loop runs forever - how do I fix an infinite loop?",Logic Error

6. "My function returns None instead of the expected value?",Logic Error

7. "My array filtering gives wrong results - off by one error help?",Logic Error

8. "Why does my comparison operator (<= vs. ==) not work as expected?",Logic Error

9. "How can I make my database query run faster?",Optimization

10. "My function takes too long - how to improve time complexity?",Optimization

11. "What's the difference between a list and a tuple in python?",Conceptual Question

12. "When should I use recursion vs. iteration?",Conceptual Question

13. "How does garbage collection work in Java?",Conceptual Question

14. "What's the difference between REST API and GraphQL?",Conceptual Question

The screenshot shows a Jupyter Notebook cell containing Python code for evaluating zero-shot, one-shot, and few-shot accuracy across three test sets: TEST_QUESTIONS, TEST_SHOT, and TEST_FEW_SHOT. The code prints accuracy values and identifies the best technique for each set.

```
# prompting_technique_evaluation.py

def main():
    for t, (query, expected) in enumerate(TEST_QUESTIONS):
        print(f"\nQuery {t+1}: {query}")
        print(f"Expected: {expected}")
        print(f"\tZero-Shot: {zero_shot_results[t][0]} predicted: {['000001'][0]} if zero_shot_results[t][0] == expected")
        print(f"\tOne-Shot: {one_shot_results[t][0]} predicted: {['000002'][0]} if one_shot_results[t][0] == expected")
        print(f"\tFew-Shot: {few_shot_results[t][0]} predicted: {['000003'][0]} if few_shot_results[t][0] == expected")

    print("\nZero-Shot Accuracy")
    print(f"Zero-Shot: {> 75} %")
    print("Accuracy Comparison (None)")
    print(f"Zero-Shot: {0} %")
    print(f"\tTechnique: (0) [Correct]: {0} / Total: {0} | Accuracy: {0} %")
    print(f"\tTechnique: (1) [Correct]: {0} / Total: {0} | Accuracy: {0} %")

    zero_correct = sum(1 for r in zero_shot_results if r[0] == 'correct')
    one_correct = sum(1 for r in one_shot_results if r[0] == 'correct')
    few_correct = sum(1 for r in few_shot_results if r[0] == 'correct')

    print(f"\tZero-Shot: {zero_correct}/{len(TEST_QUESTIONS)} | zero_shot_accuracy: {zero_shot_accuracy:.1f}%")
    print(f"\tOne-Shot: {one_correct}/{len(TEST_SHOT)} | one_shot_accuracy: {one_shot_accuracy:.1f}%")
    print(f"\tFew-Shot: {few_correct}/{len(TEST_FEW_SHOT)} | few_shot_accuracy: {few_shot_accuracy:.1f}%")

    print("\nConclusion")
    print(f"Conclusion: {> 75} %")
    print("Conclusion")
    print(f"Conclusion: {0} %")

    test_accuracy = max(zero_shot_accuracy, one_shot_accuracy, few_shot_accuracy)
    best_technique = []
    if zero_shot_accuracy == test_accuracy:
        best_technique.append("Zero-Shot")
    if one_shot_accuracy == test_accuracy:
        best_technique.append("One-Shot")
    if few_shot_accuracy == test_accuracy:
        best_technique.append("Few-Shot")

    print(f"\nIntelligent Accuracy: {test_accuracy:.1f} % | {join(best_technique)}")
```

The right side of the screen shows a "Programming help chatbot queries" interface with a sidebar for "New Agent". It lists "You are an Evaluate", "Task", and "Real-world applications". Under "Real-world applications", it says "With zero-shot or subtle queries, fine-tuned models output better Zero-Shot by learning patterns from multiple examples". It also lists two bullet points: "One-Shot can help when queries lack explicit keywords" and "Zero-Shot works well when queries have clear patterns". Below that is a "Recommendations" section with a note about using zero-shot for efficiency and a warning about accuracy on ambiguous queries. A "Complete evaluation" button is visible at the bottom.

Social Media Post Categorization

Scenario:

A social media analytics tool must classify posts into Promotion,

Complaint, Appreciation, or Inquiry.

Tasks:

1. Prepare sample social media posts.
 2. Use Zero-shot prompting.
 3. Use One-shot prompting.
 4. Use Few-shot prompting.
 5. Analyze informal language handling.

i want prompt for vsc code in python

Social media post classifier evaluation

```
social_media_post_classifier_evaluation.py
```

SOCIAL MEDIA POST CLASSIFIER - PROMPTING TECHNIQUES EVALUATION

Compares zero-shot, one-shot, and few-shot prompting methods.

import os

```
# import libraries for data preparation
prompt_prepare_data = ...
```

You are an AI assistant.

Task:

```
# Create sample social media posts for analysis.
```

Categories:

1. Description
2. Complaint
3. Appreciation
4. Reply

Requirements:

- Create at least 2 posts per category
- Posts should include informal and casual language
- Output the result in a table format with columns:
- Post ID | Post Content | Category

Test cases and expected categories:

```
test_posts = [
```

Post 1: Launched our new dinner collection! Get 50% off on all items today only, don't miss out!
Reply: Waited 30 mins for my order and it came wrong. Customer service is terrible here. So frustrating to be the last person to get food from their best service ever and the latte was perfect. You go guys anyone know if this place is open on Sundays? Need to grab something urgent?
Post 2: New menu items dropping this Friday! Come try our special burger combo. First 50 customers get a free drink with purchase.
Reply: Thank you so much for the quick response to my issue! Your support team is amazing. Problem solved. Quick question - do you guys have vegan options? Planning to visit with a friend who's vegan."

Sample social media posts for analysis

14 rows

prompt_prepare_data = ...

You are an AI assistant.

social_media_post_classifier_evaluation.py

Checking for coding errors and running the code

Running cell 0

```
# File Edit Selection View Cell Help
social_media_posts_classifier_main.py X
social_media_posts_classifier_main.py 1 of 10 1 of 2 files > Details Load file
social_media_posts_classifier_main.py
  - output only the category name
  - do not explain your answer

Post:
  "Why is this app crashing again? So annoying!"

prompt_low_shot = """
You are a social media post classifier.

task:
Classify the following social media post into one of the categories:
- Promotional
- Complaint
- Appreciation
- Inquiry

Example:
Post: "Love this new update! Works perfectly!"
Category: Appreciation

Now classify the following post...

Notes:
- Output only the category name
- Do not explain your answer

Post:
  "When will the next sale start?"

prompt_low_shot = """
You are a social media post classifier.

task:
Classify the following social media post into one of the categories:
- Promotional
- Complaint
- Appreciation
- Inquiry

Example:
Post: "Check out our latest discounts, grab them now!"
Category: Promotional

Post: "This service is terrible, totally disappointed."
Category: Complaint

Post: "Amazing support team, thanks a lot!"
Category: Appreciation

Post: "Can someone tell me how to reset my password?"
Category: Inquiry

Now classify the following post...

Notes:
- Output only the category name
- Do not explain your answer

Post:
  "Not my order today, super happy with the quality!"
```

Sample social media posts for analysis
None → Add file

prompt_low_shot = """
You are a social media post classifier.

The output now includes:
• prompt_low_shot — template for generating sample posts
• prompt_low_shot — one-shot classification example
• prompt_low_shot — one-shot classification example

The example post "When will the next sale start?" will be classified as Inquiry because it contains "when will".

All changes are complete and there are no pending errors. The script is ready to use.

Load All Keep All social_media_posts.dat social_media_posts_low_shot.dat

```
# File Edit Selection View Cell Help
social_media_posts_classifier_main.py X
social_media_posts_classifier_main.py 1 of 17 1 of 2 files > Details Load file
social_media_posts_classifier_main.py
  - output only the category name
  - do not explain your answer

Post:
  "When will the next sale start?"

prompt_low_shot = """
You are a social media post classifier.

task:
Classify the following social media post into one of the categories:
- Promotional
- Complaint
- Appreciation
- Inquiry

Example:
Post: "Check out our latest discounts, grab them now!"
Category: Promotional

Post: "This service is terrible, totally disappointed."
Category: Complaint

Post: "Amazing support team, thanks a lot!"
Category: Appreciation

Post: "Can someone tell me how to reset my password?"
Category: Inquiry

Now classify the following post...

Notes:
- Output only the category name
- Do not explain your answer

Post:
  "Not my order today, super happy with the quality!"
```

Sample social media posts for analysis
None → Add file

prompt_low_shot = """
You are a social media post classifier.

The output now includes:
• prompt_low_shot — template for generating sample posts
• prompt_low_shot — one-shot classification example
• prompt_low_shot — one-shot classification example
• prompt_low_shot — one-shot classification example

The example post "Not my order today, super happy with the quality!" will be classified as Appreciation because it contains "happy" and "quality".

All changes are complete with no pending errors. The script is ready to use.

Load All Keep All social_media_posts.dat social_media_posts_low_shot.dat

The screenshot shows a Jupyter Notebook interface with several cells of Python code. The code defines a function `analyze_results` that prints a message and then iterates over a list of posts. Each post is analyzed using a function `analyze_text_posts`, which takes a category and a list of posts. The code then prints the results. A sidebar on the right contains a sample of social media posts and a list of generated tasks:

```
def analyze_results():
    print("Starting analysis...")
    for category in categories:
        print(f"\n{category} posts: {len(posts[category])}")
        for post in posts[category]:
            print(post)

def analyze_text_posts(category, posts):
    print(f"\nAnalyzing {category} posts: {len(posts)}")
    for post in posts:
        print(post)

def analyze_informal_language():
    print("\nAnalyzing informal language patterns in social media posts using the analysis test posts...")
    for post in posts["informal"]:
        print(post)

def main():
    print("\nStarting social media post classifier analysis...")
    print("Collecting prompting techniques on informal social media posts")
    print("Starting classification analysis...")
    print("Starting classification analysis...")

    analyze_results()
    analyze_text_posts("informal", posts["informal"])
    analyze_informal_language()

    print("\nEvaluating the classifier's performance on informal language test posts...")
    test_set_results = evaluate_technique(
        zero_shot_classification,
        zero_shot_distracted,
        zero_shot_distracted,
        analyze_text_posts,
        analyze_expected_categories
    )

    print(test_set_results)
```

Sample social media posts for analysis:
You are an AI evaluator.
Analyze the following posts:
1. "Great service every, totally blew up 🎉"
2. "Big sale live now, don't miss out!!"
3. "Wow, great job guys ❤️"
4. "Any idea when this feature will be back?"

analyze_expected_categories = [
 "Complaint",
 "Promotion",
 "Appreciation",
 "Inquiry"]

analyze_informal_language():

Analyze how well different prompting techniques handle informal language in social media posts using the analysis test posts...

main():

Starting social media post classifier analysis...
Collecting prompting techniques on informal social media posts
Starting classification analysis...
Starting classification analysis...
Starting classification analysis...
Evaluating the classifier's performance on informal language test posts...
analyze_results - evaluate_technique(
 zero_shot_classification,
 zero_shot_distracted,
 zero_shot_distracted,
 analyze_text_posts,
 analyze_expected_categories