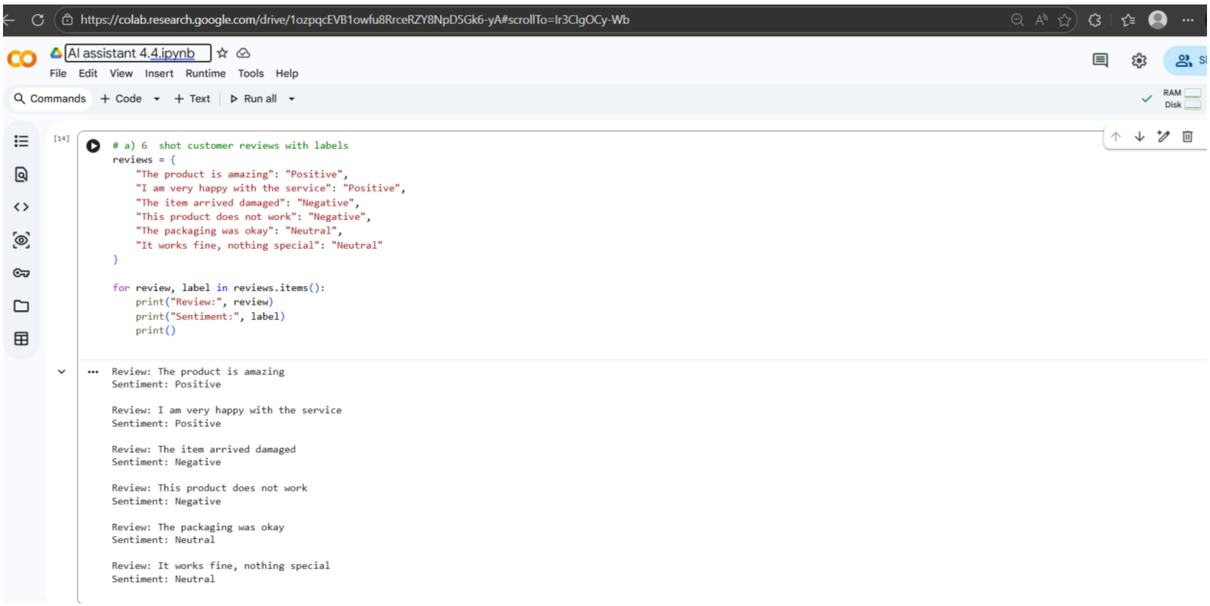
Assignment-4.4

Name:B.Sanjana

H.T.NO:2303A52306

Scenario1: E-commerce

a) 6 short customer reviews mapped to sentiment labels.



```python
# a) 6  shot customer reviews with labels
reviews = {
    "The product is amazing": "Positive",
    "I am very happy with the service": "Positive",
    "The item arrived damaged": "Negative",
    "This product does not work": "Negative",
    "The packaging was okay": "Neutral",
    "It works fine, nothing special": "Neutral"
}

for review, label in reviews.items():
    print("Review:", review)
    print("Sentiment:", label)
    print()
```

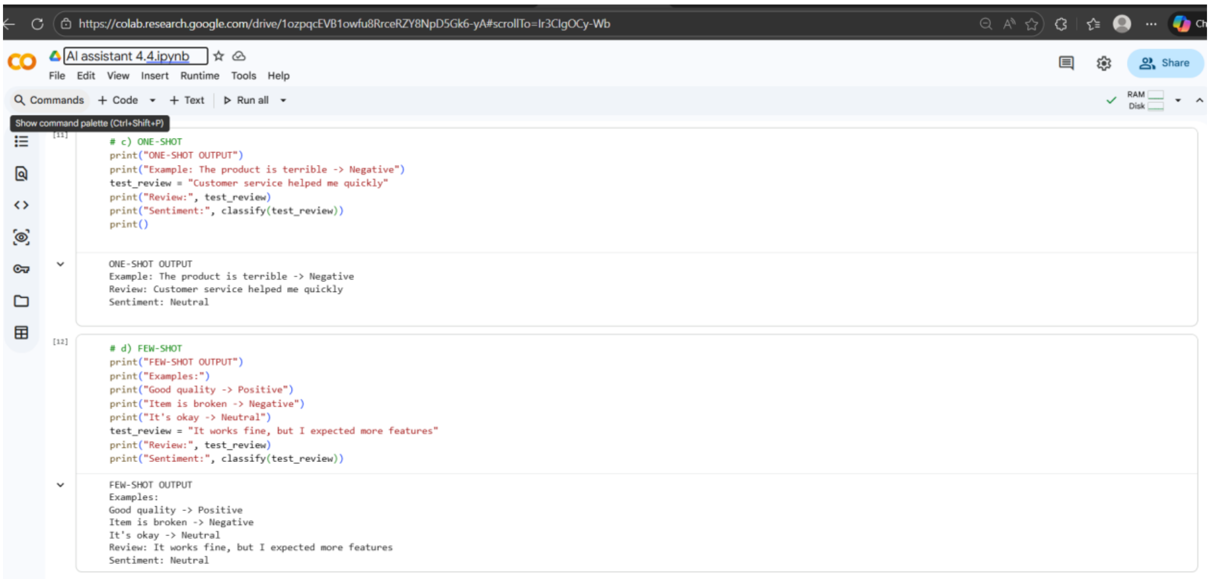
```
Review: The product is amazing
Sentiment: Positive

Review: I am very happy with the service
Sentiment: Positive

Review: The item arrived damaged
Sentiment: Negative

Review: This product does not work
Sentiment: Negative

Review: The packaging was okay
Sentiment: Neutral

Review: It works fine, nothing special
Sentiment: Neutral
```

b) Zero shot

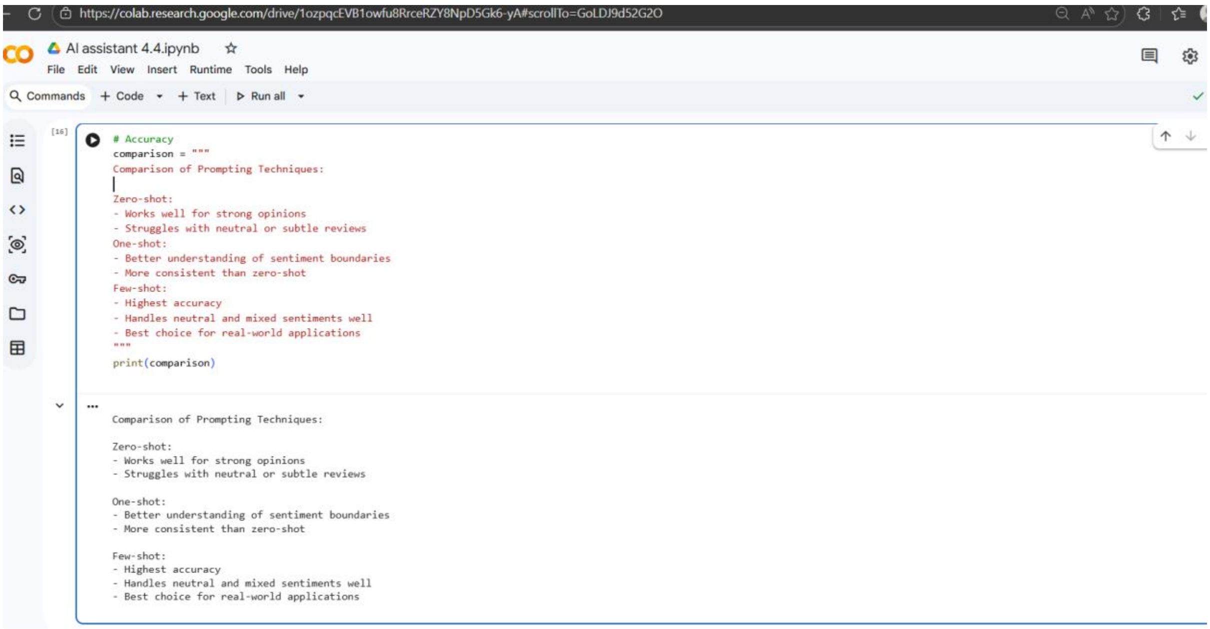
```python
# b) ZERO-SHOT
def classify(review):
    zero_shot_prompt = f"""Classify the sentiment of the following customer review into Positive, Negative, or Neutral.
Review: "{review}"
Sentiment:"""

    if "amazing" in review.lower() or "fast" in review.lower() or "love" in review.lower() or "fantastic" in review.lower():
        return "Positive"
    elif "terrible" in review.lower() or "disappointing" in review.lower() or "broke" in review.lower() or "late" in review.lower():
        return "Negative"
    else:
        return "Neutral"

print("ZERO-SHOT OUTPUT")
test_review = "The product quality is amazing and delivery was fast"
print("Review:", test_review)
print("Sentiment:", classify(test_review))
print()
```

```
ZERO-SHOT OUTPUT
Review: The product quality is amazing and delivery was fast
Sentiment: Positive
```

One-shot & Few shot



Comparision



Scenario2: Email Priority Classification

```python
# 1. Sample emails with priority labels
emails = {
    "Server is down, please fix immediately": "High",
    "Need password reset urgently": "High",
    "Can you send the meeting schedule?": "Medium",
    "Please review my project by tomorrow": "Medium",
    "Thank you for your support": "Low",
    "Just checking in to say hello": "Low"
}
high_words = ["urgent", "immediately", "down", "fix"]
medium_words = ["meeting", "review", "schedule", "tomorrow"]
# Simple classifier
def classify_email(email):
    email = email.lower()

    for word in high_words:
        if word in email:
            return "High Priority"

    for word in medium_words:
        if word in email:
            return "Medium Priority"

    return "Low Priority"

for email in emails:
    print("Email:", email)
    print("Predicted Priority:", classify_email(email))
    print("Actual Priority:", emails[email])
    print()
```

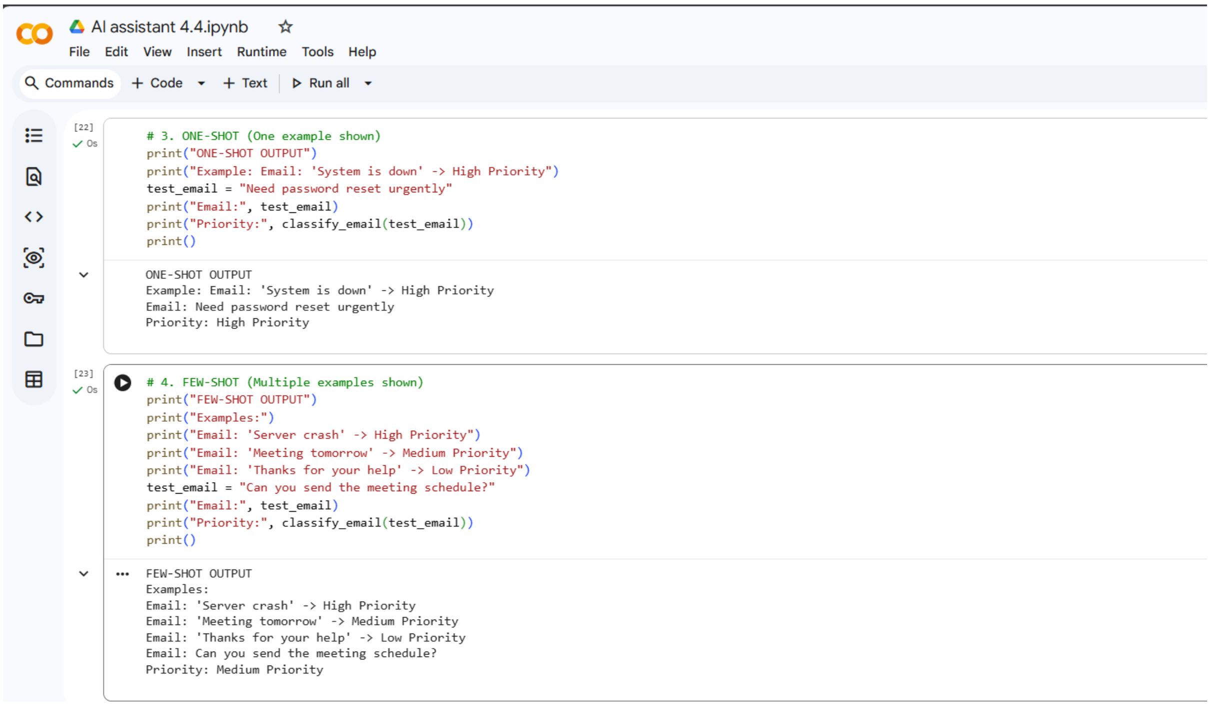
```
Email: Server is down, please fix immediately
Predicted Priority: High Priority
Actual Priority: High

Email: Need password reset urgently
Predicted Priority: High Priority
Actual Priority: High

Email: Can you send the meeting schedule?
Predicted Priority: Medium Priority
Actual Priority: Medium

Email: Please review my project by tomorrow
Predicted Priority: Medium Priority
Actual Priority: Medium

Email: Thank you for your support
Predicted Priority: Low Priority
Actual Priority: Low

Email: Just checking in to say hello
Predicted Priority: Low Priority
Actual Priority: Low
```

Zero shot

```python
# 2. ZERO-SHOT
print("ZERO-SHOT OUTPUT")
test_email = "Server is down, please fix immediately"
print("Email:", test_email)
print("Priority:", classify_email(test_email))
print()
```

```
ZERO-SHOT OUTPUT
Email: Server is down, please fix immediately
Priority: High Priority
```

One shot&few shot



AI assistant 4.4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[22]

```python
# 3. ONE-SHOT (One example shown)
print("ONE-SHOT OUTPUT")
print("Example: Email: 'System is down' -> High Priority")
test_email = "Need password reset urgently"
print("Email:", test_email)
print("Priority:", classify_email(test_email))
print()
```

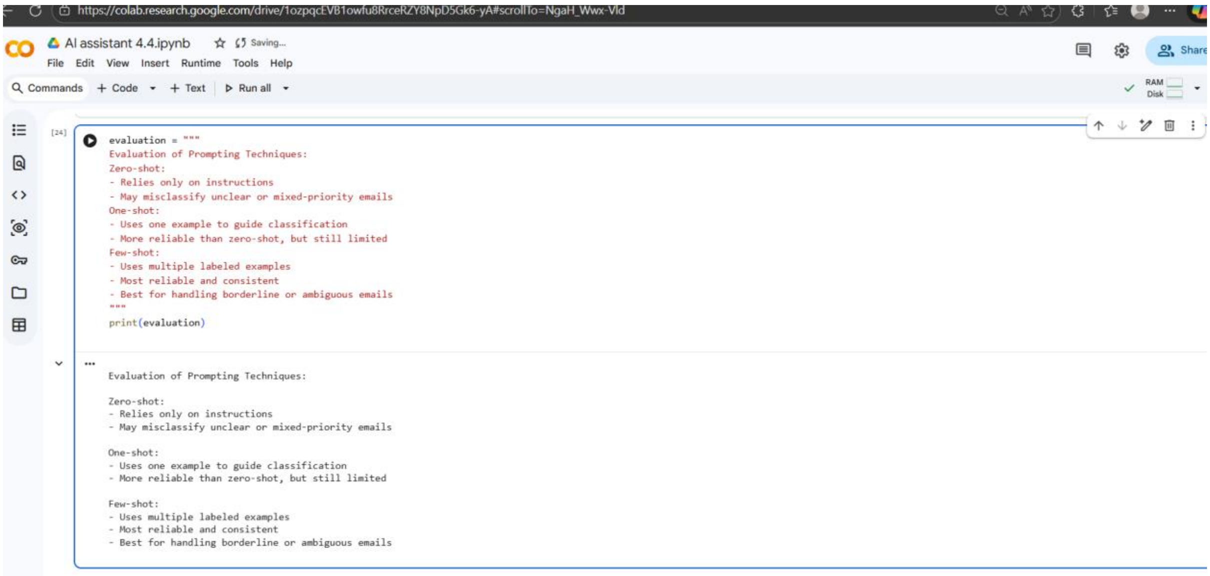
```
ONE-SHOT OUTPUT
Example: Email: 'System is down' -> High Priority
Email: Need password reset urgently
Priority: High Priority
```

[23]

```python
# 4. FEW-SHOT (Multiple examples shown)
print("FEW-SHOT OUTPUT")
print("Examples:")
print("Email: 'Server crash' -> High Priority")
print("Email: 'Meeting tomorrow' -> Medium Priority")
print("Email: 'Thanks for your help' -> Low Priority")
test_email = "Can you send the meeting schedule?"
print("Email:", test_email)
print("Priority:", classify_email(test_email))
print()
```

```
FEW-SHOT OUTPUT
Examples:
Email: 'Server crash' -> High Priority
Email: 'Meeting tomorrow' -> Medium Priority
Email: 'Thanks for your help' -> Low Priority
Email: Can you send the meeting schedule?
Priority: Medium Priority
```

Comparision



https://colab.research.google.com/drive/1ozpqcEVB1owfu8RrceRZY8NpD5Gk6-yA#scrollTo=NgaH_Wwx-Vld

AI assistant 4.4.ipynb Saving...

File Edit View Insert Runtime Tools Help

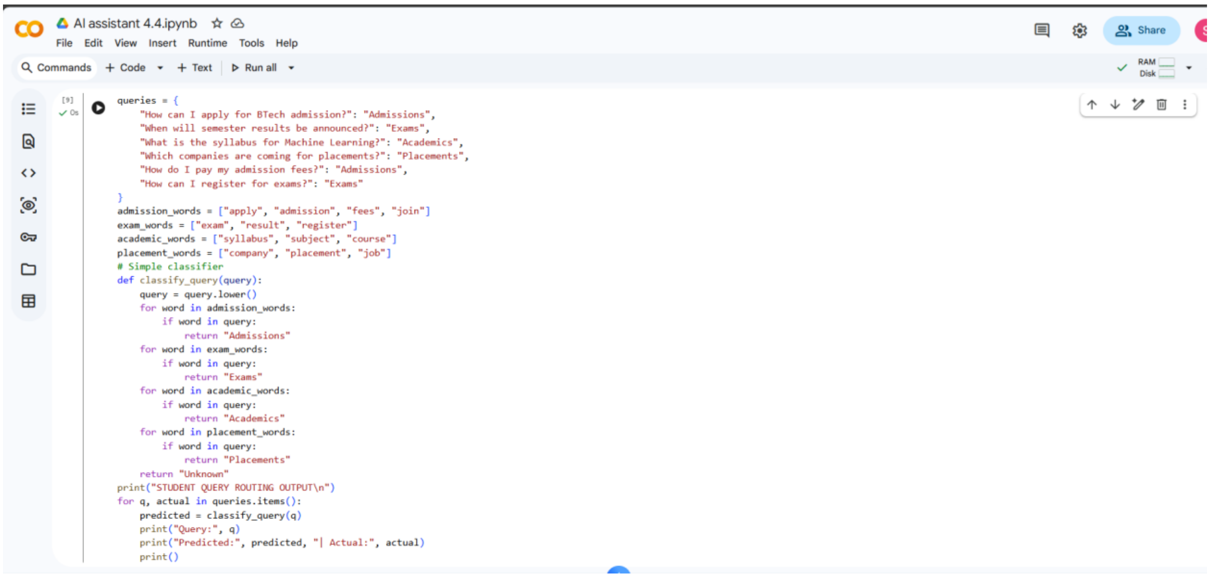
Commands + Code + Text Run all

[24]

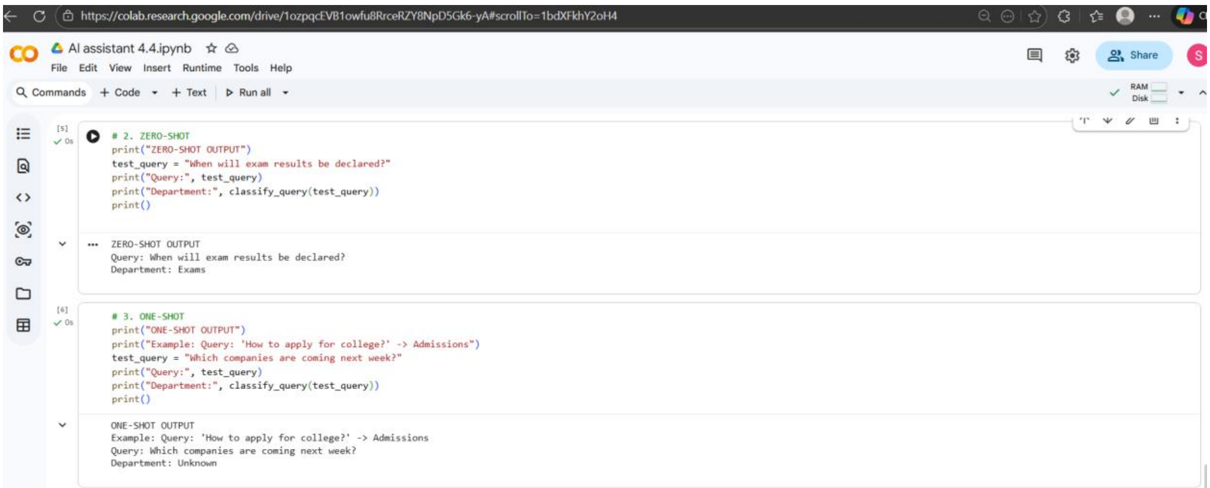
```python
evaluation = """
Evaluation of Prompting Techniques:
Zero-shot:
- Relies only on instructions
- May misclassify unclear or mixed-priority emails
One-shot:
- Uses one example to guide classification
- More reliable than zero-shot, but still limited
Few-shot:
- Uses multiple labeled examples
- Most reliable and consistent
- Best for handling borderline or ambiguous emails
"""
print(evaluation)
```

```
Evaluation of Prompting Techniques:

Zero-shot:
- Relies only on instructions
- May misclassify unclear or mixed-priority emails

One-shot:
- Uses one example to guide classification
- More reliable than zero-shot, but still limited

Few-shot:
- Uses multiple labeled examples
- Most reliable and consistent
- Best for handling borderline or ambiguous emails
```

Scenario3: Student Query Routing System



Zero shot&one shot



Few shot

AI assistant 4.4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[7]

```python
# 4. FEW-SHOT
print("FEW-SHOT OUTPUT")
print("Examples:")
print("Query: 'Exam registration' -> Exams")
print("Query: 'Syllabus for AI' -> Academics")
print("Query: 'Placement drive info' -> Placements")
test_query = "How do I pay my admission fees?"
print("Query:", test_query)
print("Department:", classify_query(test_query))
print()
```

```
FEW-SHOT OUTPUT
Examples:
Query: 'Exam registration' -> Exams
Query: 'Syllabus for AI' -> Academics
Query: 'Placement drive info' -> Placements
Query: How do I pay my admission fees?
Department: Admissions
```

[8]

```python
# 5. Accuracy Check
correct = 0
total = len(queries)

print("ACCURACY CHECK")
for q, label in queries.items():
```

https://colab.research.google.com/drive/1ozpqcEVB1owfu8RrceRZY8NpD5Gk6-yA#scrollTo=H5BQ0K8f3FPV

AI assistant 4.4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share

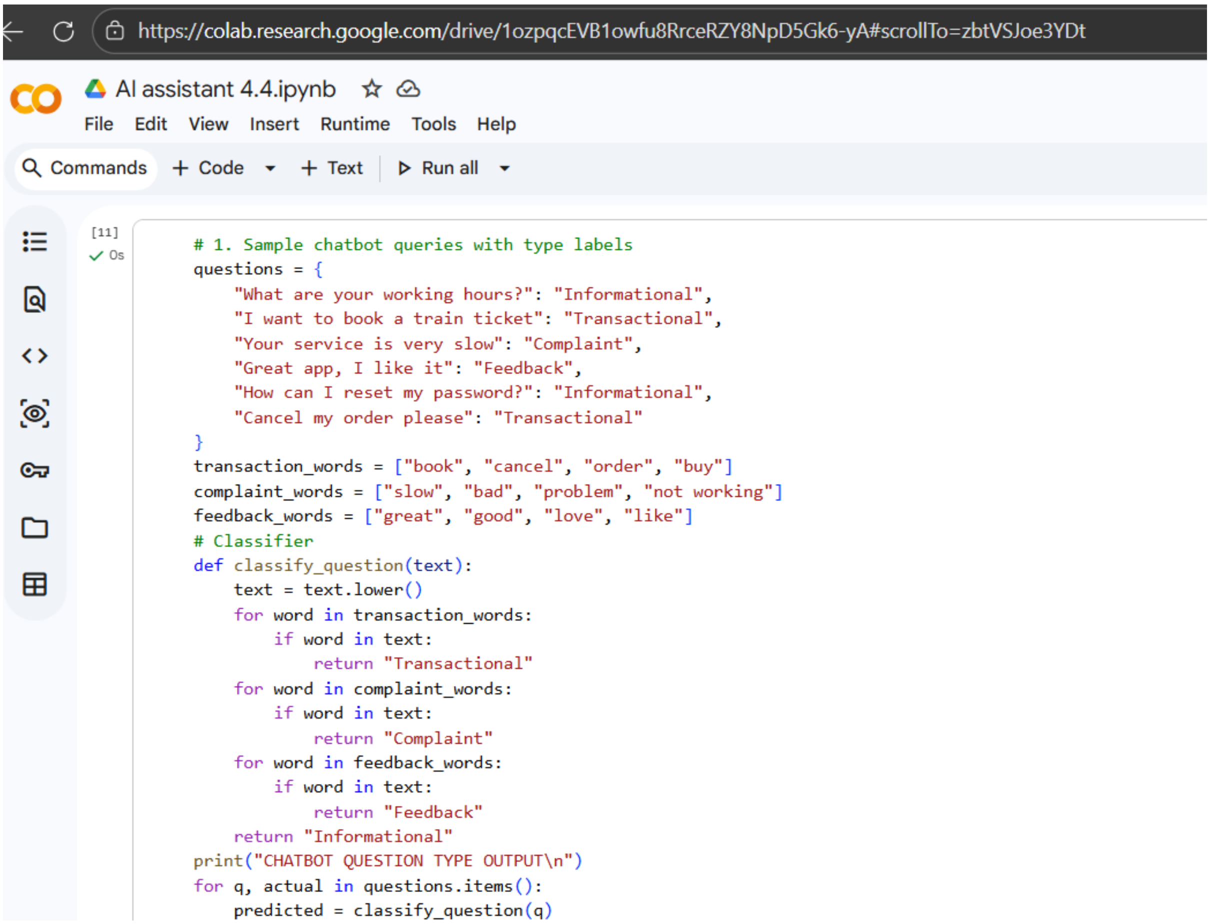
RAM Disk

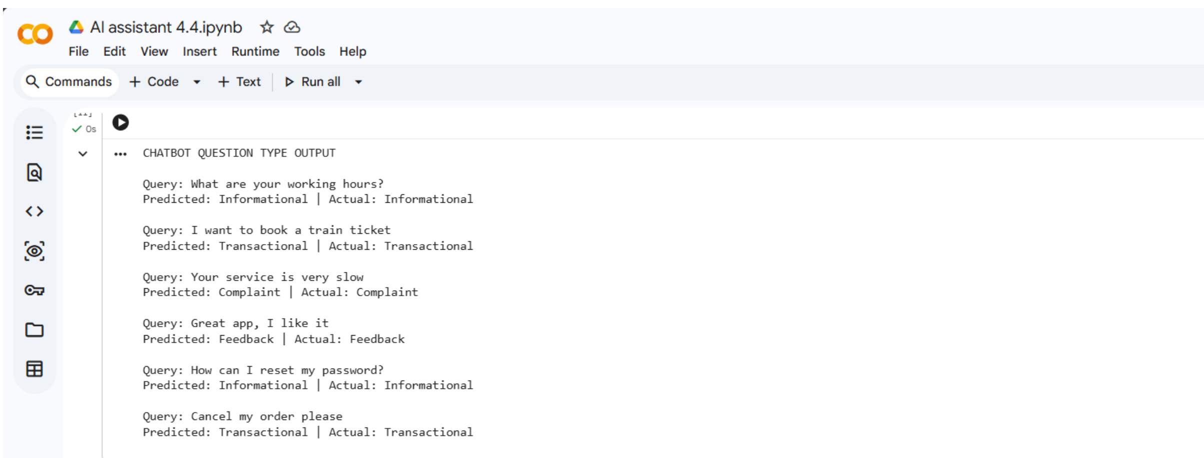
[8]

```python
correct = 0
total = len(queries)
print("ACCURACY CHECK")
for q, label in queries.items():
    result = classify_query(q)
    print("Query:", q)
    print("Predicted:", result, "| Actual:", label)
    print()
    if result == label:
        correct += 1
accuracy = (correct / total) * 100
print("Accuracy:", accuracy, "%")
print("\nConclusion: Few-shot > One-shot > Zero-shot in reliability")
```

```
ACCURACY CHECK
Query: How can I apply for BTech admission?
Predicted: Admissions | Actual: Admissions

Query: When will semester results be announced?
Predicted: Exams | Actual: Exams

Query: What is the syllabus for Machine Learning?
Predicted: Academics | Actual: Academics

Query: Which companies are coming for placements?
Predicted: Placements | Actual: Placements

Query: How do I pay my admission fees?
Predicted: Admissions | Actual: Admissions

Query: How can I register for exams?
Predicted: Exams | Actual: Exams

Accuracy: 100.0 %

Conclusion: Few-shot > One-shot > Zero-shot in reliability
```

Scenario4: Chatbot Question Type Detection





zero shot&one shot

```python
# 2. ZERO-SHOT
print("ZERO-SHOT OUTPUT")
test_q = "Cancel my ticket"
print("Query:", test_q)
print("Type:", classify_question(test_q))
print()
```

```
ZERO-SHOT OUTPUT
Query: Cancel my ticket
Type: Transactional
```

```python
# 3. ONE-SHOT
print("ONE-SHOT OUTPUT")
print("Example: Query: 'This app is bad' -> Complaint")
test_q = "I want to book a bus"
print("Query:", test_q)
print("Type:", classify_question(test_q))
print()
```

```
ONE-SHOT OUTPUT
Example: Query: 'This app is bad' -> Complaint
Query: I want to book a bus
Type: Transactional
```

Few shot

```python
# 4. FEW-SHOT
print("FEW-SHOT OUTPUT")
print("Examples:")
print("Query: 'What is your refund policy?' -> Informational")
print("Query: 'Buy a movie ticket' -> Transactional")
print("Query: 'App crashes' -> Complaint")
print("Query: 'Love this service' -> Feedback")
test_q = "Your service is not working"
print("Query:", test_q)
print("Type:", classify_question(test_q))
print()
```

```
FEW-SHOT OUTPUT
Examples:
Query: 'What is your refund policy?' -> Informational
Query: 'Buy a movie ticket' -> Transactional
Query: 'App crashes' -> Complaint
Query: 'Love this service' -> Feedback
Query: Your service is not working
Type: Complaint
```

```python
correct = 0
total = len(questions)
print("ACCURACY CHECK")
for q, label in questions.items():
    result = classify_question(q)
    print("Query:", q)
    print("Predicted:", result, "| Actual:", label)
    print()
    if result == label:
        correct += 1
accuracy = (correct / total) * 100
print("Accuracy:", accuracy, "%")
print("\nConclusion: Few-shot > One-shot > Zero-shot in reliability")
```

```
ACCURACY CHECK
Query: What are your working hours?
Predicted: Informational | Actual: Informational

Query: I want to book a train ticket
Predicted: Transactional | Actual: Transactional

Query: Your service is very slow
Predicted: Complaint | Actual: Complaint

Query: Great app, I like it
Predicted: Feedback | Actual: Feedback

Query: How can I reset my password?
Predicted: Informational | Actual: Informational

Query: Cancel my order please
Predicted: Transactional | Actual: Transactional

Accuracy: 100.0 %

Conclusion: Few-shot > One-shot > Zero-shot in reliability
```

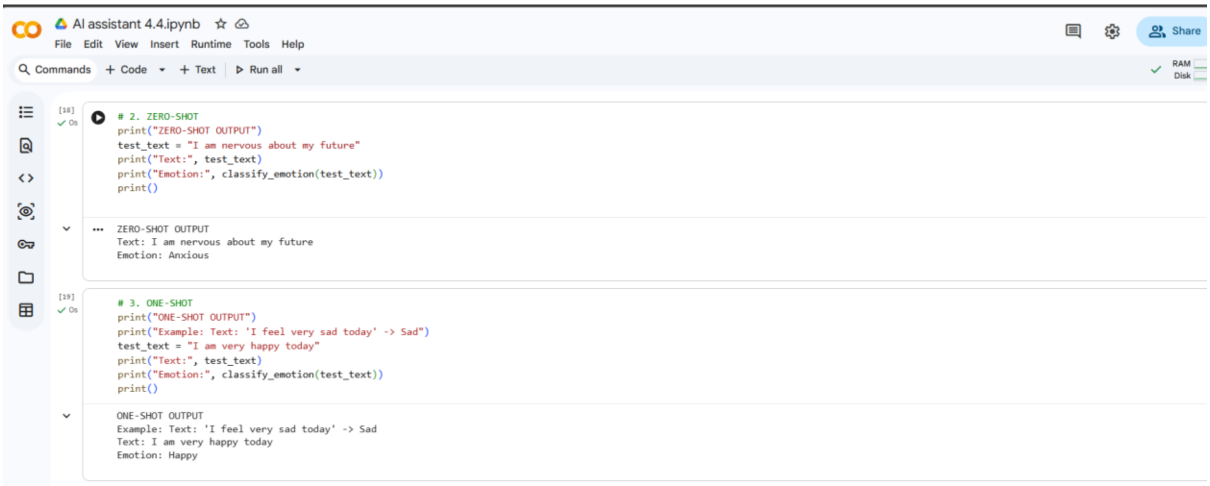
Scenario5: Emotion Detection in Text

```python
# 1. Sample texts with emotion labels
emotions = {
    "I am feeling very happy today": "Happy",
    "I feel so lonely": "Sad",
    "This makes me very angry": "Angry",
    "I am worried about my exams": "Anxious",
    "I am going to college now": "Neutral"
}
happy_words = ["happy", "great", "excited"]
sad_words = ["lonely", "sad", "cry"]
angry_words = ["angry", "mad", "furious"]
anxious_words = ["worried", "nervous", "scared"]
def classify_emotion(text):
    text = text.lower()
    for word in happy_words:
        if word in text:
          (variable) word: str
    for word in sad_words:
        if word in text:
            return "Sad"
    for word in angry_words:
        if word in text:
            return "Angry"
    for word in anxious_words:
        if word in text:
            return "Anxious"
    return "Neutral"
print("EMOTION DETECTION OUTPUT\n")
for text, actual in emotions.items():
    predicted = classify_emotion(text)
    print("Text:", text)
    print("Predicted:", predicted, "| Actual:", actual)
    print()
```

```python
    for word in sad_words:
        if word in text:
            return "Sad"
    for word in angry_words:
        if word in text:
            return "Angry"
    for word in anxious_words:
        if word in text:
            return "Anxious"
    return "Neutral"
print("EMOTION DETECTION OUTPUT\n")
for text, actual in emotions.items():
    predicted = classify_emotion(text)
    print("Text:", text)
    print("Predicted:", predicted, "| Actual:", actual)
    print()
```

```
EMOTION DETECTION OUTPUT

Text: I am feeling very happy today
Predicted: Happy | Actual: Happy

Text: I feel so lonely
Predicted: Sad | Actual: Sad

Text: This makes me very angry
Predicted: Angry | Actual: Angry

Text: I am worried about my exams
Predicted: Anxious | Actual: Anxious

Text: I am going to college now
Predicted: Neutral | Actual: Neutral
```

Zero&one shot



Few shot