Demo Use case 1 - SQLite Summarization

LLM workflow

This lab walks you through a hands-on exercise using SQLite and OpenAI
to create concise summaries from support
transcripts. It includes a sample dataset, step-by-step commands, expected outputs, validation checks, troubleshooting tips, and extension tasks you can try.

Use this in a local Python environment. The lab assumes you have a key
and permissions to create local tables and call OpenAI.
SQLite built-in.

Learning objectives

1. Load a small set of text documents (support transcripts) into SQLite.
2. Call OpenAI to produce concise summaries.
3. Inspect, evaluate, and refine summarization results.
4. Build a simple table of summaries for use in downstream workflows (agent notes, dashboards).
5. Practice troubleshooting common issues.

Environment setup (one-time)

Run these commands in a local Python environment.

```
# Install dependencies (one-time)

pip install openai python-dotenv

# Set your API key (or use .env)

export OPENAI_API_KEY="..."

# Create/open the SQLite database



sqlite3 support_tickets.db

# (Optional) verify SQLite

python -c "import sqlite3"
```

Step 1 — Create sample dataset

Create a table of customer support transcripts and populate it with example rows.

```
CREATE TABLE customer_support_tickets (
ticket_id INTEGER,
created_at TEXT,
customer_name TEXT,
language TEXT,
category TEXT,
transcript TEXT,
summary TEXT);

INSERT INTO customer_support_tickets (ticket_id, created_at, customer_name,
language, category, transcript) VALUES
(1001, CURRENT_TIMESTAMP(), 'Alice Brown', 'English', 'Delivery',
 'The product was delayed by three days. I did not receive any shipping updates. I tried
calling support twice and email but no response. I need it for a birthday.'),
(1002, CURRENT_TIMESTAMP(), 'Rajesh Kumar', 'Hindi', 'Payment',
 'मेरा भुगतान विफल हो गया लेकिन मेरे खाते से पैसे कट गए। मैंने बैंक से भी पुष्टि कर ली है। कृपया पैसे
वापस भेजें।'),
(1003, CURRENT_TIMESTAMP(), 'Maria Lopez', 'Spanish', 'Product',
 'El producto llegó con la pantalla rota. Necesito un reemplazo o reembolso. Adjunto
fotos.'),
```

```
(1004, CURRENT_TIMESTAMP(), 'John Smith', 'English', 'Refund',
 'I requested a refund two weeks ago and the refund has not been processed yet. The order number is ORD-54321. Can you confirm the status?'),
(1005, CURRENT_TIMESTAMP(), 'Sophie Dubois', 'French', 'Quality',
 'Le produit ne correspond pas à la description sur le site. Les dimensions sont incorrectes et la finition est mauvaise.');
```

Step 2 — Basic summarization (single column)

After Python summarization, view results with SQL.

```
SELECT
  ticket_id,
  customer_name,
  language,
  category,
summary AS auto_summary
FROM customer_support_tickets
ORDER BY ticket_id;
```

What to expect

A result set with an auto_summary column where each transcript has a concise summary. Example expected outputs (samples, wording may vary):

ticket_id	summary
1001	Customer reports a three day delivery delay and lack of shipping updates; attempted contact with support.
1002	Payment failed but money was deducted from account; customer requests refund.
1003	Product arrived with a broken screen; customer requests replacement or refund.
1004	Refund requested two weeks ago for order ORD-54321; customer requests status update.
1005	Product does not match website description; incorrect dimensions and poor finish.

Note: Exact phrasing can vary because the summarization model may produce slightly different wording.

Step 3 — Summarization with context columns

Sometimes you want the summary to include metadata, for example language or ticket id. Use || to include extra context in the input text so the model can incorporate it into the summary.

```
SELECT
  ticket_id,
context_text =
('Ticket ' || ticket_id || ' [' || language || '] Transcript: ' || transcript)
AS summary_with_context
FROM customer_support_tickets
```

```
ORDER BY ticket_id;
```

Why do this

Including small structured context helps produce summaries that mention the ticket id or language when desirable, useful for auditability or downstream automation.

Step 4 — Shorter or longer summaries (technique)

OpenAI summaries are concise by default. If you need a stricter length target or a specific format (for example, a one-line bullet or JSON), use OPENAI.COMPLETE with an instruction prompt.

Example of constrained output using OPENAI.COMPLETE:

COMPLETE:

```
SELECT
  ticket_id,
OPENAI.COMPLETE(
'gpt-4o-mini',
    CONCAT(
      'Produce a one-sentence summary in English for the following transcript. ',
      'Return only the sentence. Transcript: ', transcript
    )
  ) AS one_sentence_summary
FROM customer_support_tickets;
```

Note: COMPLETE gives more control over instructions and format. Use it must enforce exact output structure.

Step 5 — Write summaries back to a table for reporting

Store generated summaries in a new table for downstream reports, dashboards, or agent interfaces.

```
CREATE TABLE ticket_summaries AS
SELECT
  ticket_id,
  customer_name,
  language,
  category,
summary AS summary,
  CURRENT_TIMESTAMP() AS summary_generated_at
FROM customer_support_tickets;
```

Validate inserted rows:

```
SELECT * FROM ticket_summaries ORDER BY ticket_id;
```

Step 6 — Evaluation and quality checks

Create a simple quality framework to check summarization outputs.

1. Sample verification

- Manually inspect 5 to 10 random summaries for faithfulness and usefulness.

2. Automated checks

- Check for empty summaries

```
SELECT COUNT(*) AS empty_count
FROM ticket_summaries
WHERE TRIM(summary) = '';
```

- Check for summaries that repeat the whole input (indicating poor summarization)

```
SELECT ticket_id
FROM ticket_summaries s
JOIN customer_support_tickets t USING (ticket_id)
WHERE LENGTH(s.summary) > LENGTH(t.transcript) * 0.8;
```

3. Accuracy sampling

- Create a small human-labelled dataset (ground truth) of 50 transcripts with reference summaries and compute simple overlap metrics (ROUGE-like or manual scoring).

Suggested manual scoring rubric (for human reviewers)

- Score 0: Incorrect or misleading summary

- Score 1: Partial, misses critical point
- Score 2: Accurate and concise

Log results in a table for tracking model quality over time.