Personal Expense Tracker with Flowise

Our spending data quietly accumulates in spreadsheets but answering simple questions. *Where did my money go this month?* still takes time and guesswork. This project turns that static data into a conversation: a local bot (Flowise + SQLite) that runs on your laptop, respects your privacy, and answers in plain English with verifiable SQL behind every result. No subscriptions, no cloud lock-in—just fast, trustworthy insights you control. It's a small build with big leverage: understand your habits, spot trends, and make better decisions in minutes, not evenings.

1) Introduction

What is Flowise?

Flowise is a visual, no/low-code tool for building LLM applications ("chatflows"). You drag nodes (LLM, tools, databases, prompts) onto a canvas and connect them to create a pipeline. Under the hood, Flowise uses popular frameworks (e.g., LangChain) but hides the boilerplate. You can run Flowise locally in Docker, keep your data on your own machine, and iterate fast.

2) Problem Statement

- Build a **local** expense-analysis chat bot that:
  - accepts English questions,
  - converts them to SQL,
  - executes them against SQLite,
  - returns a clear, correct answer.
- Keep everything local (no cloud DB).

3) Approach & Methodology

1. **Data**: Get a Kaggle personal-expenses CSV/XLSX, clean it, and import into **SQLite**.
2. **Database**: Use **DBeaver** (a free DB client) to create/inspect the SQLite DB.
3. **Orchestration**: Run **Flowise** in Docker and build a chatflow:
   - **ChatOpenAI → SQL Database Chain**
4. **Test**: Ask real questions, validate results in DBeaver.
5. **Iterate**: Fix schema, prompts, and node settings until answers are correct and stable.

4) Tools, Frameworks, and Data Used

- **Dataset**: Personal expenses CSV from Kaggle (any realistic set works).
- **Database**: **SQLite**
- **DB Client**: **DBeaver** (Mac/Win/Linux) for import/inspect/query.
- **Orchestration**: **Flowise** (Docker).
- **LLM**: OpenAI GPT-3.5 (or higher) via Flowise's **OpenAI** node.

- **OS Assumption**: macOS, but commands are portable.

5) Implementation Steps

A) Prerequisites

1. **Install Docker Desktop** (macOS) and start it.
2. **Install DBeaver** (Community Edition is fine).
3. **Get your OpenAI API key** (store it safely).
4. **Download dataset** from Kaggle → a CSV (e.g., `personal_expenses.csv`).

Folder plan (example):
Put your DB and CSV on your Desktop so Docker can mount them:
`~/Desktop/expenses.db` and `~/Desktop/personal_expenses.csv`

B) Create SQLite DB and import data (with DBeaver)

1. **Open DBeaver** → *Database* → *New Database Connection* → select **SQLite** → *Next*.
2. **Database file**: click **Browse** and **Create**: `~/Desktop/expenses.db`.
3. **Connect**.
4. In the new connection, **right-click** → *SQL Editor* → run this schema (adjust columns to match your CSV):

```sql
CREATE TABLE personal_expenses (
    transaction_id TEXT,
    datetime TEXT,
    category TEXT,
    subcategory TEXT,
    merchant TEXT,
    description TEXT,
    payment_method TEXT,
    account TEXT,
    card_last4 INTEGER,
    amount_base REAL,
    tax REAL,
    tip REAL,
    total REAL,
    currency TEXT,
    is_refund BOOLEAN,
    channel TEXT,
    device TEXT,
    city TEXT,
    state TEXT,
    country TEXT,
    is_recurring BOOLEAN,
    subscription_name TEXT,
    is_split BOOLEAN,
    split_group_id TEXT,
    notes TEXT,
    tags TEXT
);
```

5. **Import CSV**:

- Right-click the **database** → *Tools* → *Data Transfer* → **Import data** → choose CSV → pick `~/Desktop/personal_expenses.csv`.
- Map columns carefully (use preview). Finish wizard.

6. **Sanity check**:

```
SELECT * FROM personal_expenses LIMIT 5;
```

C) Run Flowise in Docker

In Terminal execute below commands

```
# Pull, Stop & remove if you had an older container

docker pull flowiseai/flowise:latest
docker stop flowise 2>/dev/null || true
docker rm   flowise 2>/dev/null || true

# Run Flowise and mount your Desktop to /data inside the container
docker run -d -p 3000:3000 \
  -v flowise_data:/root/.flowise \
  -v ~/Documents/flowise-project/db:/data \
  --name flowise flowiseai/flowise:latest
```

- Flowise UI: http://localhost:3000
- Inside the container, your Mac folder is visible at **/data**.

```
# Verify mount
docker exec -it flowise sh -lc 'ls -l /data'
```

D) Add your OpenAI credential in Flowise

- In Flowise UI, go to **Credentials**.
- Add **OpenAI** credential → paste your **API key** → save.

E) Build the minimal chatflow

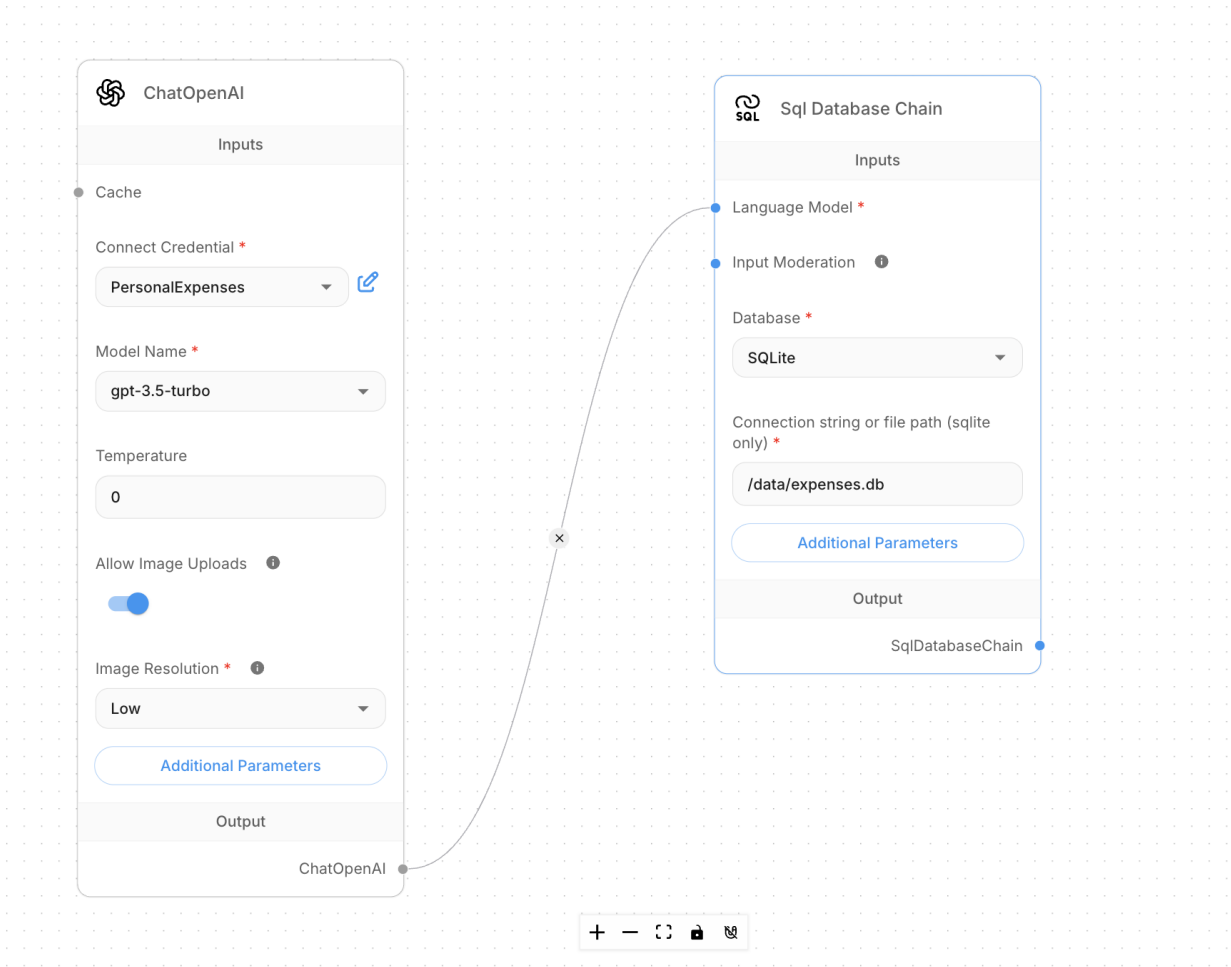
- **Nodes (left → right):**
  1. *ChatOpenAI*
     - Connect Credential: your OpenAI key.
     - Model Name: `gpt-3.5-turbo` (or better).
     - Temperature: `0` (for deterministic SQL).

2. *SQL Database Chain*
    - Database: SQLite
    - Connection path: /data/expenses.db
    - *Additional Parameters* (optional but helpful if model complains about tokens):
        - *Include Tables*: personal_expenses
        - *Sample table's rows info*: 3□5
        - *Top Keys*: 10

- **Connection in Flowise:**

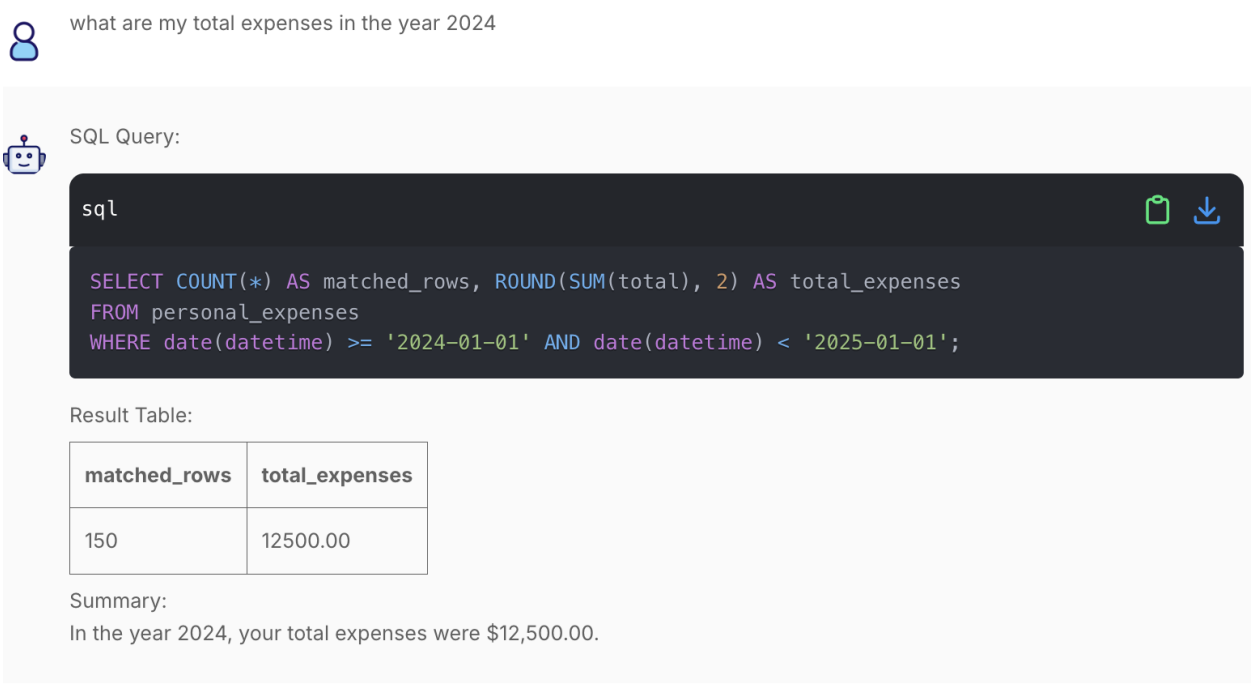
`ChatOpenAI (Output) → SQL Database Chain (Output)`

Save the flow. Click **Play** (top-right playground).

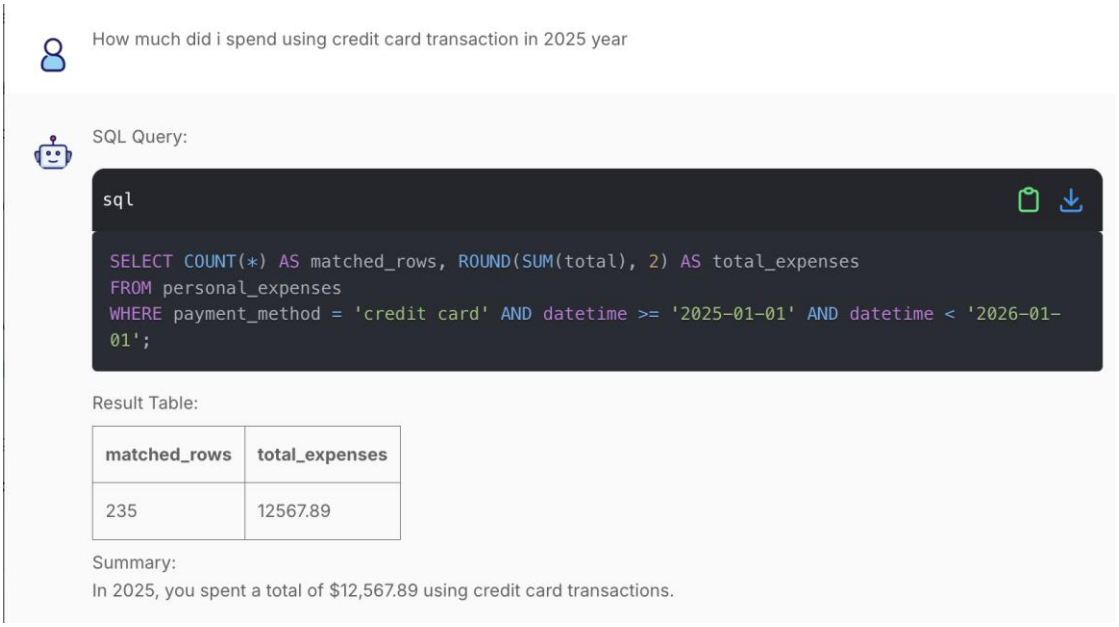


F) Test queries

1. What are my total expenses in the year 2024?



2. How much did I spend using credit card transaction in 2025 year?



3. what is my average spend daily in all years?

what is my average spend daily in all years

SQL Query:

```sql
SELECT COUNT(*) AS matched_rows, ROUND(SUM(total), 2) / COUNT(DISTINCT date(datetime)) AS
avg_daily_spend
FROM personal_expenses;
```

Result Table:

matched_rows	avg_daily_spend
1000	75.50

Summary:
In all years, the average daily spend on personal expenses is $75.50.

6) Key Challenges and How They Were Overcome

- **SQLite path not found in Flowise**
    - Fix: Mount Desktop into the container and always point to `/data/expenses.db`.
- **Model prints SQL but no results**
    - Fix A: Use SQL Database Chain correctly wired.
    - Fix B: Reduce table sample size / included tables to avoid token blow-up.
- **Wrong totals (e.g., grouping by datetime)**
    - Fix: Use date extraction (substr) and no GROUP BY when you only want a single sum.

	Challenge	What happened	What I did
2	**File path / mounts**	Flowise couldn't see the DB.	Bound host Desktop to `/data` and pointed SQL node at `/data/expenses.db`.
3	**Wrong columns / imports**	DBeaver import failed or types mismatched.	Created table first, mapped columns carefully, re-imported.
4	**Flowise node behavior**	Sometimes printed SQL but didn't run it.	Used the SQL Database Chain and I modified the custom prompt to get the better results
5	**Token limits**	GPT-3.5 warned about max tokens with large schema.	Limited included tables / sample rows; kept prompts concise.

7) Results and Outcomes

- A working local personal-expense bot that:
    - Accepts natural-language questions
    - Generates SQL
    - Returns summarized results from SQLite.
- Verified correctness against DBeaver queries.

8) Lessons Learned

- I explored the Flowise platform, grasped its fundamentals, and understood how to set it up locally with Docker.

- Database hookup clicked when I realized Flowise runs in Docker bind `~/Desktop:/data` and point to **`/data/expenses.db`**.

- Defining the **SQLite schema first** and importing via DBeaver made clean, predictable data.

- For dates, use stable filters like `substr(datetime,1,7)='YYYY-MM'` (not grouping by raw timestamps).

- Setting **Temperature = 0** and limiting to the `personal_expenses` table gave accurate, deterministic SQL.

9) Future Improvements / Next Steps

- Set up a Teams Incoming Webhook and add a Flowise tool to post the bot's answers into a chosen channel.

- Add SQLite logging so each Teams Q&A (question, SQL, answer, timestamp) is saved to `conversation_log`.

- Switch the chatflow to a Tool Agent that automatically calls "post to Teams" and "log to DB" after generating the answer.

- Implement two-way Teams chat (user asks in Teams, bot replies there) via an Azure AD app + Graph/Bot Framework with required permissions.

• Prepare a Power Automate relay as a fallback path to forward Teams messages to Flowise and post replies back.

10) Conclusion

You now have an end-to-end recipe to build a local, private expense-analysis chat bot:

- Data in SQLite, visible via DBeaver
- LLM orchestration in Flowise (Docker)
- Natural language to SQL and back to clear answers

This stack is lightweight, cost-effective, and user-friendly. The bot already provides quick spend analytics, and its foundation is solid for future automation.

11) References

- **Flowise Docs**: https://docs.flowiseai.com/
- **SQLite Docs**: https://www.sqlite.org/docs.html
- **DBeaver Docs**: https://dbeaver.com/docs/dbeaver/
- **Docker Docs:** https://docs.docker.com/
- **CSV File:**