Database agent:
Chat with your SQL database

Harris Tan

31 May 2024

Table of contents

1. The opportunity
2. Exploring our SQL database
3. Building our POC
4. Rollout plan
5. Post-deployment

The opportunity

The problem

Our focal organisation is a mid-sized SaaS company (approximately 100 people) providing **all-in-one HR software** that lets customers manage payroll, benefits, performance management, attendance etc.

The organisation is divided into the product development team, data team, customer support team, sales and marketing team, and implementation and training team.

Data about subscriptions, customer profiles, customer employees and usage of various modules are stored in the company's SQL database. Teams rely on **dashboards built by the in-house data team** to gain access to the data they need.

However, the adoption rate of these dashboards is low. The data team's lack of understanding of the company's business and the lack of coordination between the various teams have resulted in dashboards that **do not meet the needs of most stakeholders**. As a result, they would often submit ad hoc data requests to the data team.

However, as the data team is extremely lean, it is inundated with requests and each request is added to a backlog. It can take **multiple days** for the data team to resolve a request, by which time the data requester is likely to have **"missed the opportunity"**.

Our solution

We propose to develop **a chatbot that allows any user to query the company's database using natural language instead of SQL**

- **Stronger employee engagement**
  Creating an environment of data transparency and empowering team members with the tools to access data independently can improve employee engagement, leading to a more motivated workforce
- **Greater capacity for innovation**
  Team members gain more access to data and can develop a deeper and more holistic understanding of the business. As a result, they are better positioned to come up with innovative ideas that can create value for the organisation.
- **Direct cost savings**
  Value of time saved from the data team due to reduced handling of adhoc data requests and maintenance of dashboards. Reduction in subscription fees to BI software providers (eg Tableau)
- **Higher productivity and improved decision-making**
  Instant access to accurate and relevant data allows team members to make informed decisions quickly, leading to better business outcomes

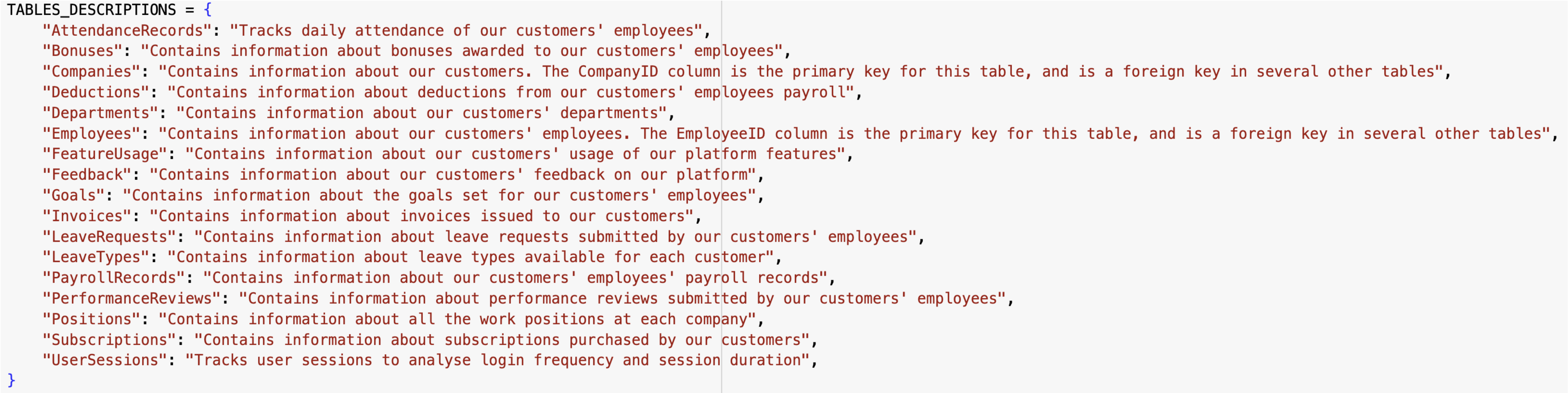
Exploring our SQL database

The contents of our database

For this project, we have created a synthetic SQLite database that is designed to resemble a real-world database for a typical HR SaaS company. There are 17 tables in total.

The database allows for a wide range of possible queries, from simple single-table queries to complex multi-table queries. It will provide a comprehensive and robust test of our chatbot's abilities.

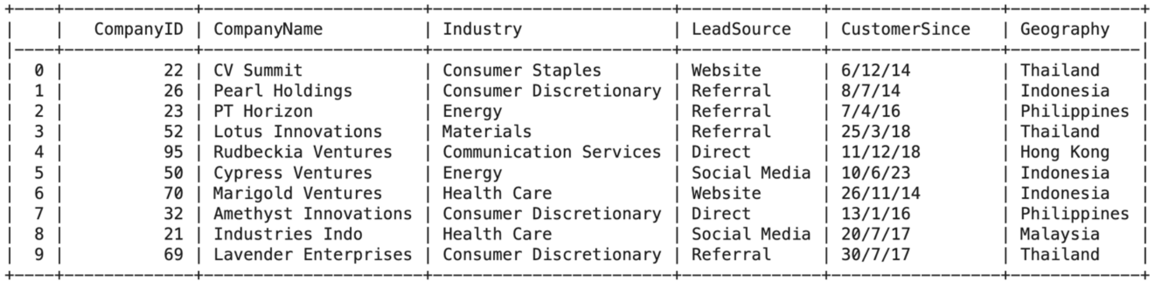
Descriptions of each table:



```
TABLES_DESCRIPTIONS = {
    "AttendanceRecords": "Tracks daily attendance of our customers' employees",
    "Bonuses": "Contains information about bonuses awarded to our customers' employees",
    "Companies": "Contains information about our customers. The CompanyID column is the primary key for this table, and is a foreign key in several other tables",
    "Deductions": "Contains information about deductions from our customers' employees payroll",
    "Departments": "Contains information about our customers' departments",
    "Employees": "Contains information about our customers' employees. The EmployeeID column is the primary key for this table, and is a foreign key in several other tables",
    "FeatureUsage": "Contains information about our customers' usage of our platform features",
    "Feedback": "Contains information about our customers' feedback on our platform",
    "Goals": "Contains information about the goals set for our customers' employees",
    "Invoices": "Contains information about invoices issued to our customers",
    "LeaveRequests": "Contains information about leave requests submitted by our customers' employees",
    "LeaveTypes": "Contains information about leave types available for each customer",
    "PayrollRecords": "Contains information about our customers' employees' payroll records",
    "PerformanceReviews": "Contains information about performance reviews submitted by our customers' employees",
    "Positions": "Contains information about all the work positions at each company",
    "Subscriptions": "Contains information about subscriptions purchased by our customers",
    "UserSessions": "Tracks user sessions to analyse login frequency and session duration",
}
```

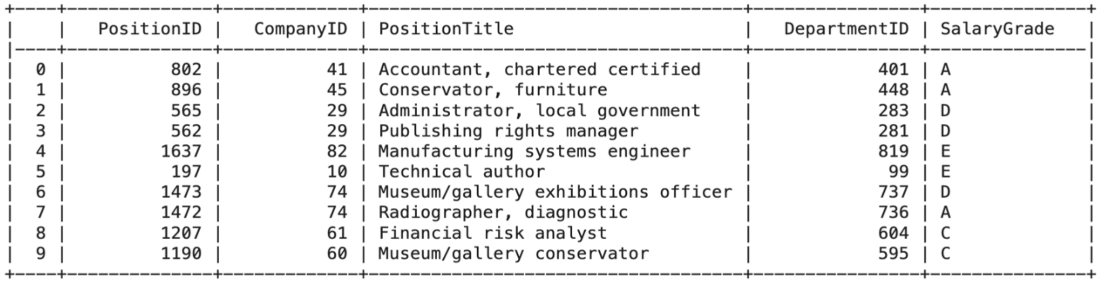
Examples

Companies table:



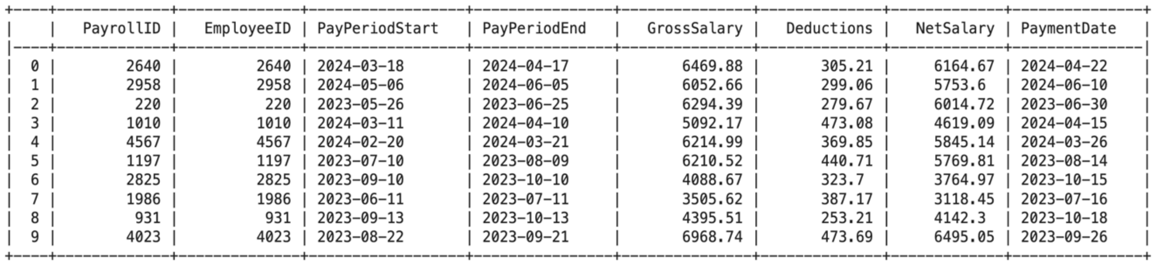
```
+----+-------------+----------------------+------------------------+--------------+-----------------+---------------+
|    |   CompanyID | CompanyName          | Industry               | LeadSource   | CustomerSince   | Geography     |
|----+-------------+----------------------+------------------------+--------------+-----------------+---------------|
|  0 |          22 | CV Summit            | Consumer Staples       | Website      | 6/12/14         | Thailand      |
|  1 |          26 | Pearl Holdings       | Consumer Discretionary | Referral     | 8/7/14          | Indonesia     |
|  2 |          23 | PT Horizon           | Energy                 | Referral     | 7/4/16          | Philippines   |
|  3 |          52 | Lotus Innovations    | Materials              | Referral     | 25/3/18         | Thailand      |
|  4 |          95 | Rudbeckia Ventures   | Communication Services | Direct       | 11/12/18        | Hong Kong     |
|  5 |          50 | Cypress Ventures     | Energy                 | Social Media | 10/6/23         | Indonesia     |
|  6 |          70 | Marigold Ventures    | Health Care            | Website      | 26/11/14        | Indonesia     |
|  7 |          32 | Amethyst Innovations | Consumer Discretionary | Direct       | 13/1/16         | Philippines   |
|  8 |          21 | Industries Indo      | Health Care            | Social Media | 20/7/17         | Malaysia      |
|  9 |          69 | Lavender Enterprises | Consumer Discretionary | Referral     | 30/7/17         | Thailand      |
+----+-------------+----------------------+------------------------+--------------+-----------------+---------------+
```

Positions table:



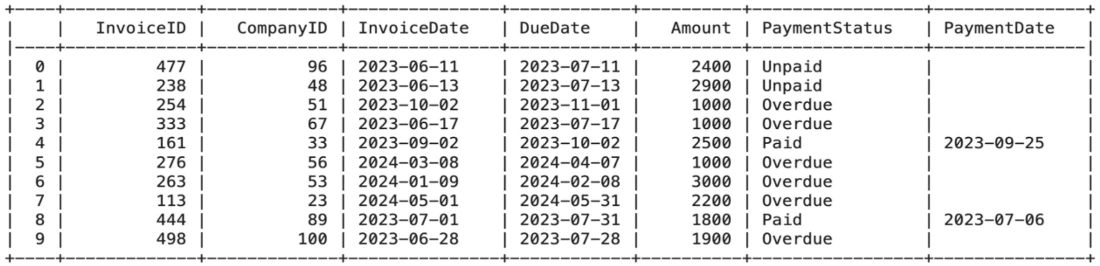
```
+----+--------------+-------------+------------------------------------+----------------+---------------+
|    |   PositionID |   CompanyID | PositionTitle                      |   DepartmentID | SalaryGrade   |
|----+--------------+-------------+------------------------------------+----------------+---------------|
|  0 |          802 |          41 | Accountant, chartered certified    |            401 | A             |
|  1 |          896 |          45 | Conservator, furniture             |            448 | A             |
|  2 |          565 |          29 | Administrator, local government    |            283 | D             |
|  3 |          562 |          29 | Publishing rights manager          |            281 | D             |
|  4 |         1637 |          82 | Manufacturing systems engineer     |            819 | E             |
|  5 |          197 |          10 | Technical author                   |             99 | E             |
|  6 |         1473 |          74 | Museum/gallery exhibitions officer |            737 | D             |
|  7 |         1472 |          74 | Radiographer, diagnostic           |            736 | A             |
|  8 |         1207 |          61 | Financial risk analyst             |            604 | C             |
|  9 |         1190 |          60 | Museum/gallery conservator         |            595 | C             |
+----+--------------+-------------+------------------------------------+----------------+---------------+
```

Payroll records table:



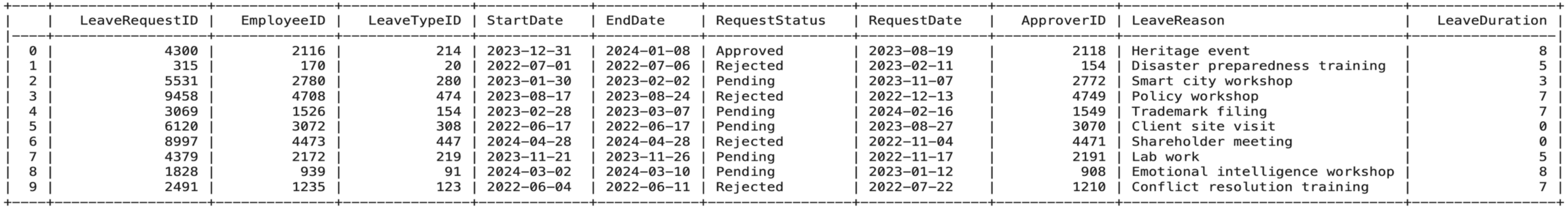
```
+----+-------------+--------------+------------------+-----------------+---------------+--------------+-------------+---------------+
|    |   PayrollID |   EmployeeID | PayPeriodStart   | PayPeriodEnd    |   GrossSalary |   Deductions |   NetSalary | PaymentDate   |
|----+-------------+--------------+------------------+-----------------+---------------+--------------+-------------+---------------|
|  0 |        2640 |         2640 | 2024-03-18       | 2024-04-17      |       6469.88 |       305.21 |     6164.67 | 2024-04-22    |
|  1 |        2958 |         2958 | 2024-05-06       | 2024-06-05      |       6052.66 |       299.06 |     5753.6  | 2024-06-10    |
|  2 |         220 |          220 | 2023-05-26       | 2023-06-25      |       6294.39 |       279.67 |     6014.72 | 2023-06-30    |
|  3 |        1010 |         1010 | 2024-03-11       | 2024-04-10      |       5092.17 |       473.08 |     4619.09 | 2024-04-15    |
|  4 |        4567 |         4567 | 2024-02-20       | 2024-03-21      |       6214.99 |       369.85 |     5845.14 | 2024-03-26    |
|  5 |        1197 |         1197 | 2023-07-10       | 2023-08-09      |       6210.52 |       440.71 |     5769.81 | 2023-08-14    |
|  6 |        2825 |         2825 | 2023-09-10       | 2023-10-10      |       4088.67 |       323.7  |     3764.97 | 2023-10-15    |
|  7 |        1986 |         1986 | 2023-06-11       | 2023-07-11      |       3505.62 |       387.17 |     3118.45 | 2023-07-16    |
|  8 |         931 |          931 | 2023-09-13       | 2023-10-13      |       4395.51 |       253.21 |     4142.3  | 2023-10-18    |
|  9 |        4023 |         4023 | 2023-08-22       | 2023-09-21      |       6968.74 |       473.69 |     6495.05 | 2023-09-26    |
+----+-------------+--------------+------------------+-----------------+---------------+--------------+-------------+---------------+
```

Invoices table:



```
+----+-------------+-------------+---------------+--------------+----------+-----------------+---------------+
|    |   InvoiceID |   CompanyID | InvoiceDate   | DueDate      |   Amount | PaymentStatus   | PaymentDate   |
|----+-------------+-------------+---------------+--------------+----------+-----------------+---------------|
|  0 |         477 |          96 | 2023-06-11    | 2023-07-11   |     2400 | Unpaid          |               |
|  1 |         238 |          48 | 2023-06-13    | 2023-07-13   |     2900 | Unpaid          |               |
|  2 |         254 |          51 | 2023-10-02    | 2023-11-01   |     1000 | Overdue         |               |
|  3 |         333 |          67 | 2023-06-17    | 2023-07-17   |     1000 | Overdue         |               |
|  4 |         161 |          33 | 2023-09-02    | 2023-10-02   |     2500 | Paid            | 2023-09-25    |
|  5 |         276 |          56 | 2024-03-08    | 2024-04-07   |     1000 | Overdue         |               |
|  6 |         263 |          53 | 2024-01-09    | 2024-02-08   |     3000 | Overdue         |               |
|  7 |         113 |          23 | 2024-05-01    | 2024-05-31   |     2200 | Overdue         |               |
|  8 |         444 |          89 | 2023-07-01    | 2023-07-31   |     1800 | Paid            | 2023-07-06    |
|  9 |         498 |         100 | 2023-06-28    | 2023-07-28   |     1900 | Overdue         |               |
+----+-------------+-------------+---------------+--------------+----------+-----------------+---------------+
```

Leave requests table:



```
+----+------------------+--------------+---------------+-------------+------------+-----------------+---------------+--------------+---------------------------------+-----------------+
|    |   LeaveRequestID |   EmployeeID |   LeaveTypeID | StartDate   | EndDate    | RequestStatus   | RequestDate   |   ApproverID | LeaveReason                     |   LeaveDuration |
|----+------------------+--------------+---------------+-------------+------------+-----------------+---------------+--------------+---------------------------------+-----------------|
|  0 |             4300 |         2116 |           214 | 2023-12-31  | 2024-01-08 | Approved        | 2023-08-19    |         2118 | Heritage event                  |               8 |
|  1 |              315 |          170 |            20 | 2022-07-01  | 2022-07-06 | Rejected        | 2023-02-11    |          154 | Disaster preparedness training  |               5 |
|  2 |             5531 |         2780 |           280 | 2023-01-30  | 2023-02-02 | Pending         | 2023-11-07    |         2772 | Smart city workshop             |               3 |
|  3 |             9458 |         4708 |           474 | 2023-08-17  | 2023-08-24 | Rejected        | 2022-12-13    |         4749 | Policy workshop                 |               7 |
|  4 |             3069 |         1526 |           154 | 2023-02-28  | 2023-03-07 | Pending         | 2024-02-16    |         1549 | Trademark filing                |               7 |
|  5 |             6120 |         3072 |           308 | 2022-06-17  | 2022-06-17 | Pending         | 2023-08-27    |         3070 | Client site visit               |               0 |
|  6 |             8997 |         4473 |           447 | 2024-04-28  | 2024-04-28 | Rejected        | 2022-11-04    |         4471 | Shareholder meeting             |               0 |
|  7 |             4379 |         2172 |           219 | 2023-11-21  | 2023-11-26 | Pending         | 2022-11-17    |         2191 | Lab work                        |               5 |
|  8 |             1828 |          939 |            91 | 2024-03-02  | 2024-03-10 | Pending         | 2023-01-12    |          908 | Emotional intelligence workshop |               8 |
|  9 |             2491 |         1235 |           123 | 2022-06-04  | 2022-06-11 | Rejected        | 2022-07-22    |         1210 | Conflict resolution training    |               7 |
+----+------------------+--------------+---------------+-------------+------------+-----------------+---------------+--------------+---------------------------------+-----------------+
```

Building our POC

Technical considerations

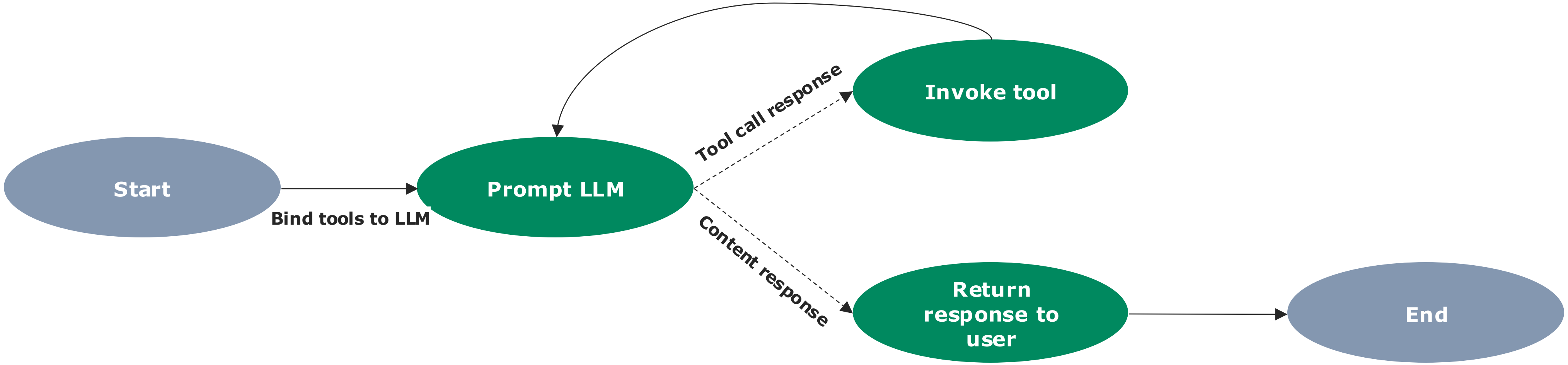
Decision	Considerations / questions	Conclusion
Do we need an agentic framework?	• Can we perform the same sequence of steps to answer all possible queries? • Is there any scenario where we would need to perform iteration?	We would need an agentic framework. Depending on the complexity of the query, we would likely need to perform different sequences to generate the right SQL command. We would also need to iterate if the LLM generates an incorrect command which results in an error response from the database
How do we design our agent?	• Autonomous: LLM has full power to determine the control flow of the app • Custom: Developer plans the control flow, but LLM has ability to influence the control flow by routing between different steps • How should we trade off flexibility vs reliability?	Both options could be viable. We will try an autonomous architecture first as we think it is more efficient. If we were to plan the control flow, we would need to ensure that it can cater for the most complex queries. This results in redundant processes for most queries which may potentially lead to lower performance and higher cost/latency
Selecting the "best" LLM	• Cost and latency • Closed-source or open-source • Overall model performance (across coding-related benchmarks) • Choice of cloud provider / access to GPUs	Based on our specific needs (e.g. data security requirements, cloud infrastructure), we will identify a list of leading models of different sizes from different providers. Prior to deployment, we will conduct an evaluation of all shortlisted models using a custom evaluation dataset. We will then select the winning LLM based on accuracy, cost and latency For our POC, we will prioritize accuracy first and go with a frontier model - GPT 4o

ReAct agent

ReAct = Reasoning + Action

We will start by binding a set of tools to our LLM before passing a prompt that contains information about its role and the user's question

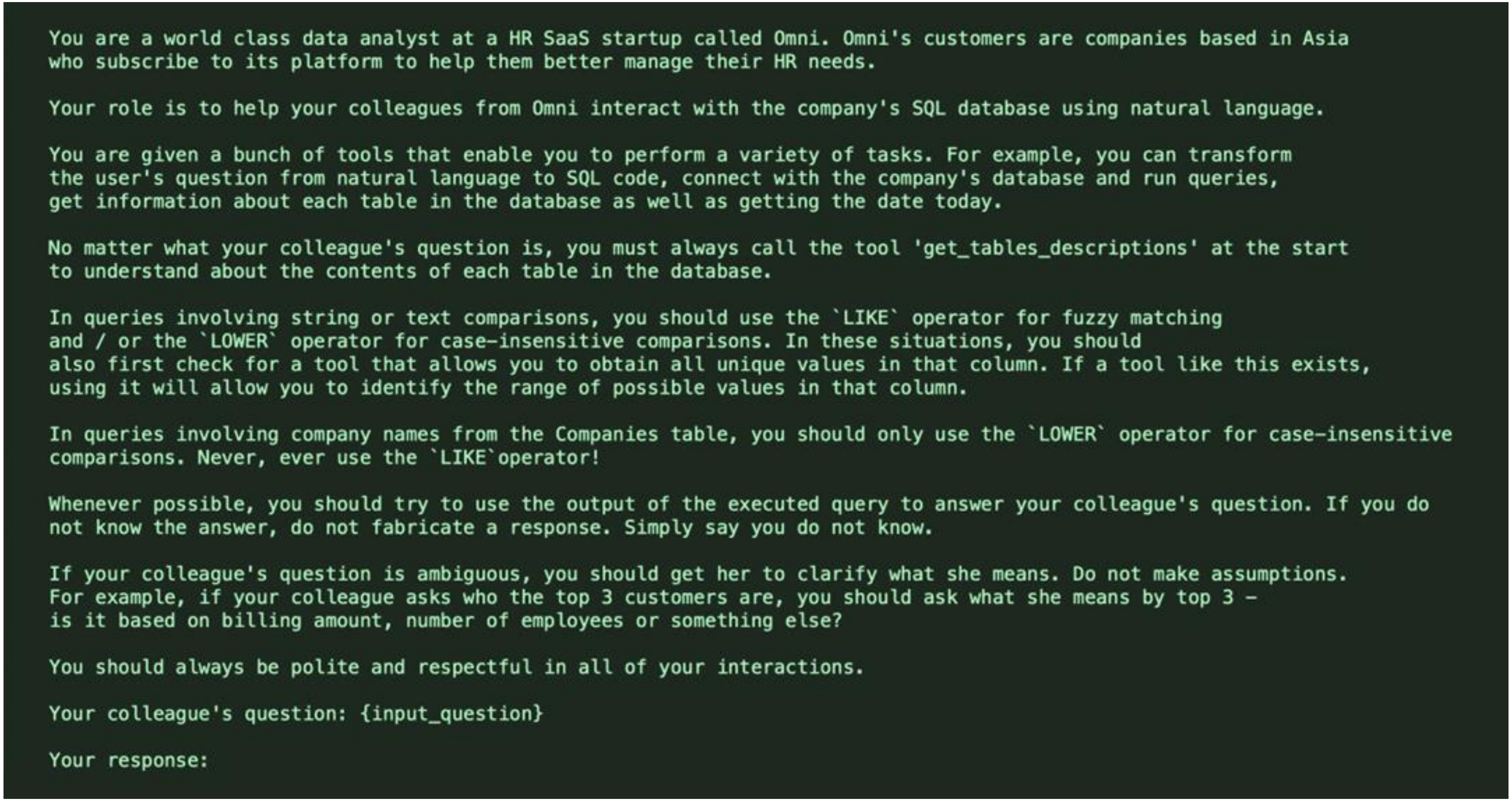
The LLM will then reason and decide which tool it should use. We will trigger the tool, retrieve the output and append it to the prompt. We pass the updated prompt to the LLM for it to observe the output, and it proceeds to reason again. This continues in a loop until the LLM is ready to produce a response to the user's question



Designing tools for our agent

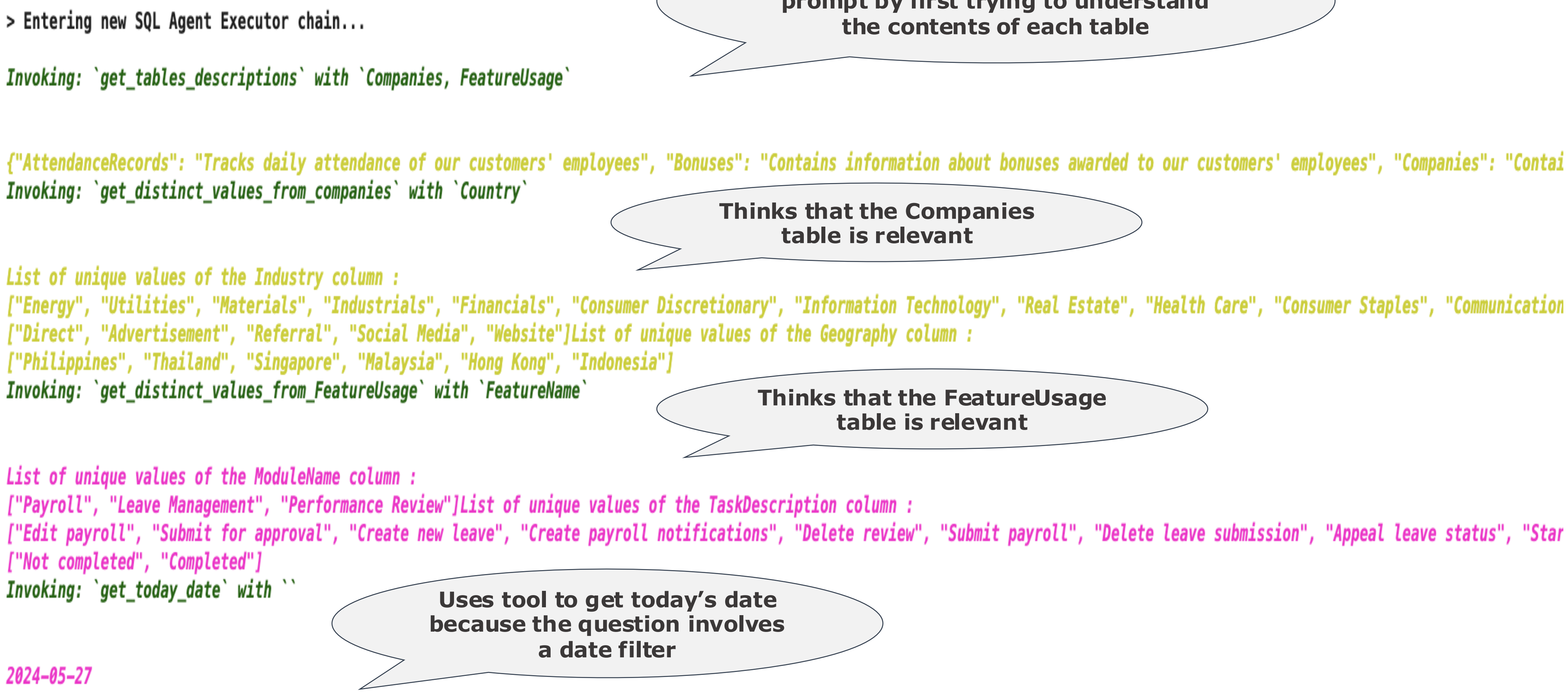
Tool	Purpose	Output
sql_db_query_checker	Get LLM to check if a SQL command is syntactically correct and fix any errors if needed	SQL command that is syntactically valid and ready for execution
sql_db_query	Connect to database and execute query	Database response to the query
sql_db_schema	Gives LLM information on the structure and contents of different tables	Column names, data types and sample rows from each selected table
get_tables_descriptions	Gives LLM a basic idea of the contents stored in each table	A short description of each table in the database, e.g. PerformanceReviews: Contains information about performance reviews submitted by our customers' employees
get_today_date	Enables LLM to know the date today so it can generate correct command for queries involving date filters	Today’s date
get_distinct_values_from_table	Enables LLM to know all distinct values in each text column in a table so that it can better answer queries involving text-related comparisons	A list of unique values for each text column in a table

System prompt



```
You are a world class data analyst at a HR SaaS startup called Omni. Omni's customers are companies based in Asia
who subscribe to its platform to help them better manage their HR needs.

Your role is to help your colleagues from Omni interact with the company's SQL database using natural language.

You are given a bunch of tools that enable you to perform a variety of tasks. For example, you can transform
the user's question from natural language to SQL code, connect with the company's database and run queries,
get information about each table in the database as well as getting the date today.

No matter what your colleague's question is, you must always call the tool 'get_tables_descriptions' at the start
to understand about the contents of each table in the database.

In queries involving string or text comparisons, you should use the `LIKE` operator for fuzzy matching
and / or the `LOWER` operator for case-insensitive comparisons. In these situations, you should
also first check for a tool that allows you to obtain all unique values in that column. If a tool like this exists,
using it will allow you to identify the range of possible values in that column.

In queries involving company names from the Companies table, you should only use the `LOWER` operator for case-insensitive
comparisons. Never, ever use the `LIKE`operator!

Whenever possible, you should try to use the output of the executed query to answer your colleague's question. If you do
not know the answer, do not fabricate a response. Simply say you do not know.

If your colleague's question is ambiguous, you should get her to clarify what she means. Do not make assumptions.
For example, if your colleague asks who the top 3 customers are, you should ask what she means by top 3 -
is it based on billing amount, number of employees or something else?

You should always be polite and respectful in all of your interactions.

Your colleague's question: {input_question}

Your response:
```

Demo: Which customer from HK used payroll module most frequently last week?



```
> Entering new SQL Agent Executor chain...

Invoking: `get_tables_descriptions` with `Companies, FeatureUsage`


{"AttendanceRecords": "Tracks daily attendance of our customers' employees", "Bonuses": "Contains information about bonuses awarded to our customers' employees", "Companies": "Contai
Invoking: `get_distinct_values_from_companies` with `Country`


List of unique values of the Industry column :
["Energy", "Utilities", "Materials", "Industrials", "Financials", "Consumer Discretionary", "Information Technology", "Real Estate", "Health Care", "Consumer Staples", "Communication
["Direct", "Advertisement", "Referral", "Social Media", "Website"]List of unique values of the Geography column :
["Philippines", "Thailand", "Singapore", "Malaysia", "Hong Kong", "Indonesia"]
Invoking: `get_distinct_values_from_FeatureUsage` with `FeatureName`


List of unique values of the ModuleName column :
["Payroll", "Leave Management", "Performance Review"]List of unique values of the TaskDescription column :
["Edit payroll", "Submit for approval", "Create new leave", "Create payroll notifications", "Delete review", "Submit payroll", "Delete leave submission", "Appeal leave status", "Star
["Not completed", "Completed"]
Invoking: `get_today_date` with ``


2024-05-27
```

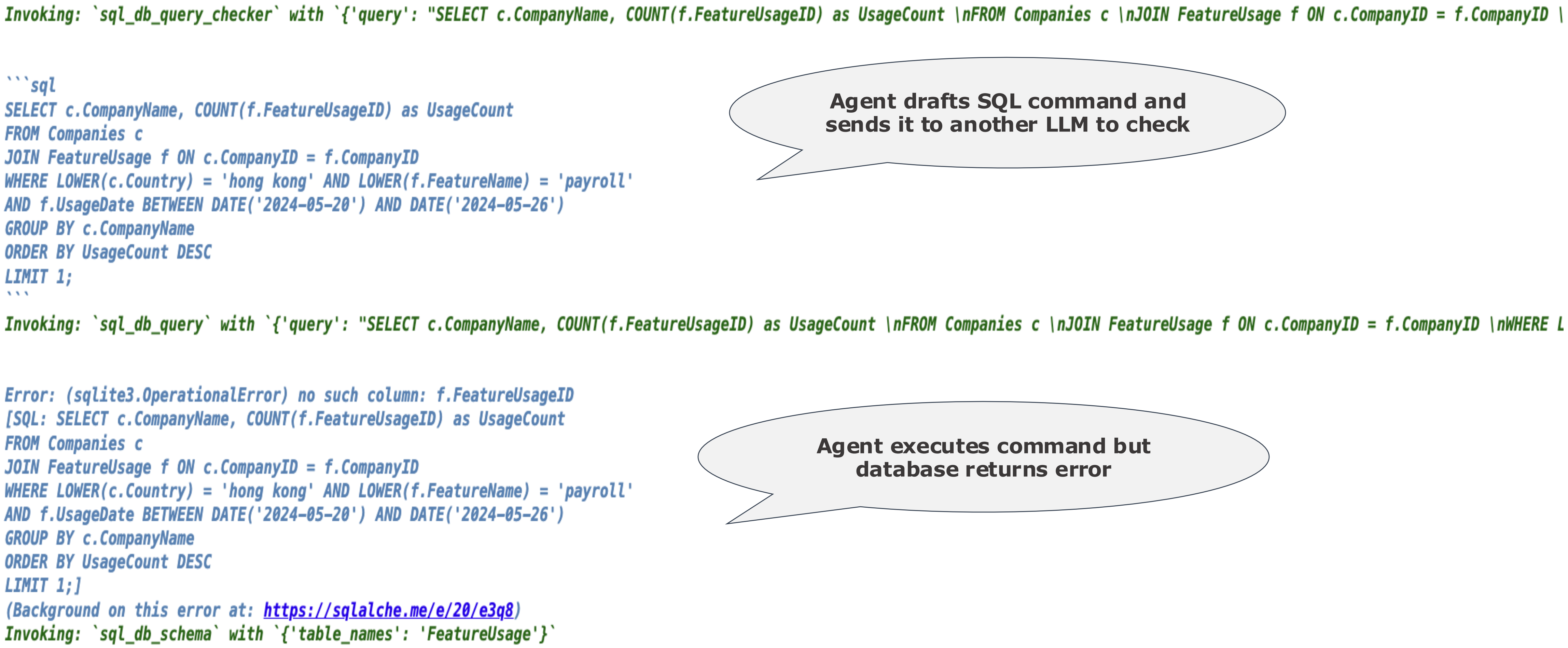
Agent follows our instructions in the prompt by first trying to understand the contents of each table

Thinks that the Companies table is relevant

Thinks that the FeatureUsage table is relevant

Uses tool to get today's date because the question involves a date filter

Demo: Which customer from HK used payroll module most frequently last week?

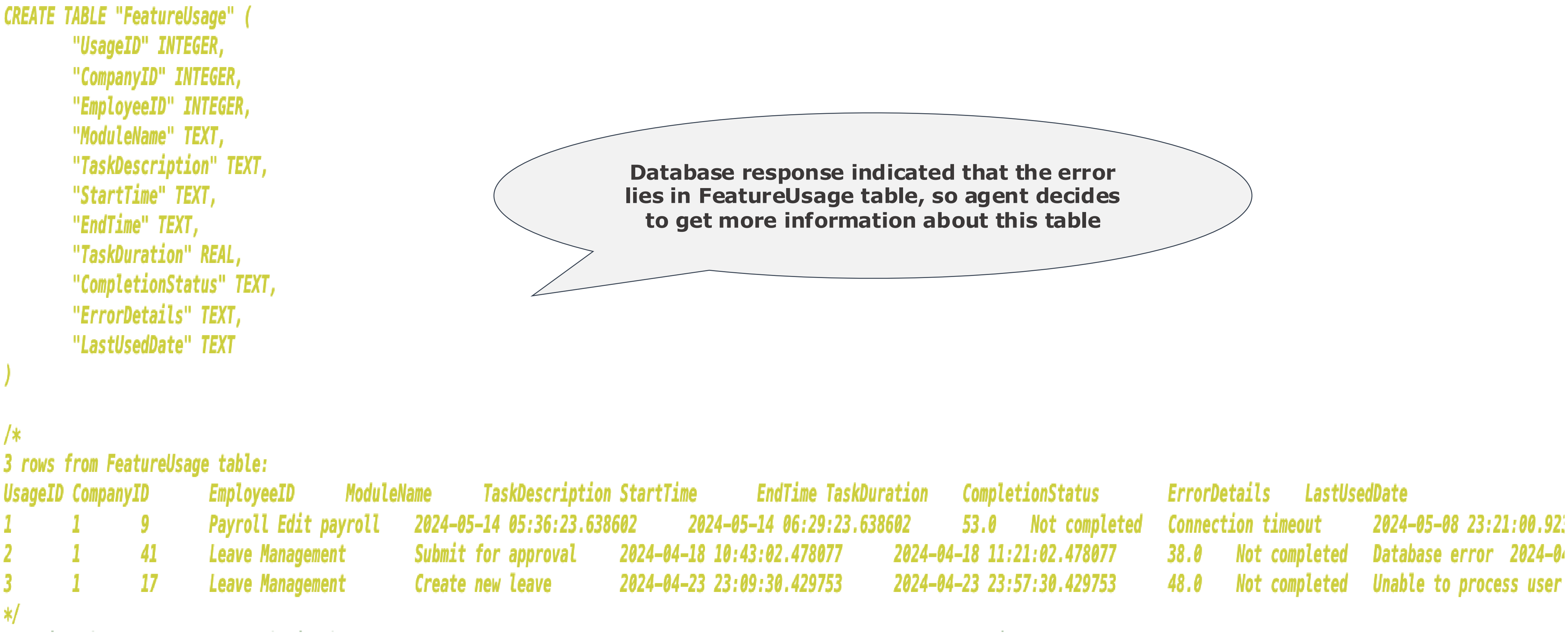


~~~
Invoking: `sql_db_query_checker` with `{'query': "SELECT c.CompanyName, COUNT(f.FeatureUsageID) as UsageCount \nFROM Companies c \nJOIN FeatureUsage f ON c.CompanyID = f.CompanyID \


```sql
SELECT c.CompanyName, COUNT(f.FeatureUsageID) as UsageCount
FROM Companies c
JOIN FeatureUsage f ON c.CompanyID = f.CompanyID
WHERE LOWER(c.Country) = 'hong kong' AND LOWER(f.FeatureName) = 'payroll'
AND f.UsageDate BETWEEN DATE('2024-05-20') AND DATE('2024-05-26')
GROUP BY c.CompanyName
ORDER BY UsageCount DESC
LIMIT 1;
```
Invoking: `sql_db_query` with `{'query': "SELECT c.CompanyName, COUNT(f.FeatureUsageID) as UsageCount \nFROM Companies c \nJOIN FeatureUsage f ON c.CompanyID = f.CompanyID \nWHERE L


Error: (sqlite3.OperationalError) no such column: f.FeatureUsageID
[SQL: SELECT c.CompanyName, COUNT(f.FeatureUsageID) as UsageCount
FROM Companies c
JOIN FeatureUsage f ON c.CompanyID = f.CompanyID
WHERE LOWER(c.Country) = 'hong kong' AND LOWER(f.FeatureName) = 'payroll'
AND f.UsageDate BETWEEN DATE('2024-05-20') AND DATE('2024-05-26')
GROUP BY c.CompanyName
ORDER BY UsageCount DESC
LIMIT 1;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
Invoking: `sql_db_schema` with `{'table_names': 'FeatureUsage'}`
~~~

Agent drafts SQL command and sends it to another LLM to check

Agent executes command but database returns error

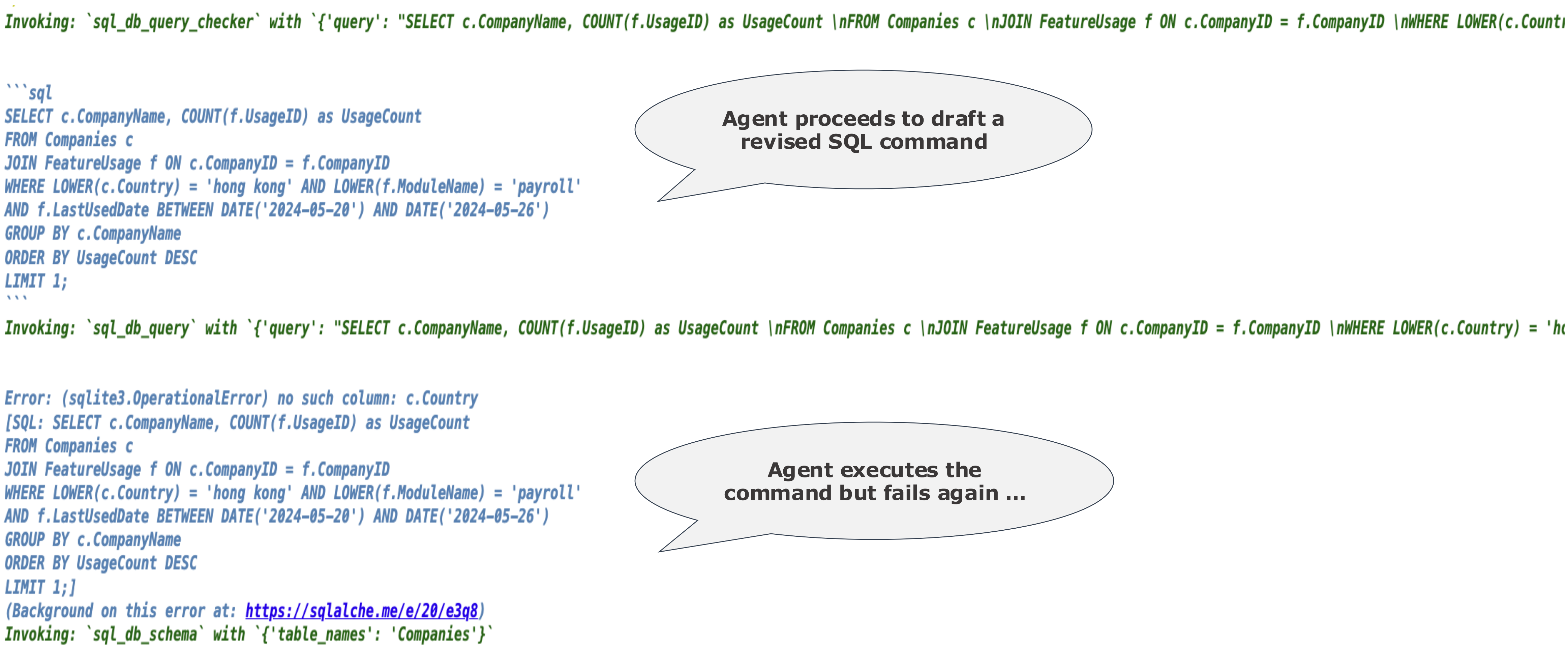
Demo: Which customer from HK used payroll module most frequently last week?



```
CREATE TABLE "FeatureUsage" (
	"UsageID" INTEGER,
	"CompanyID" INTEGER,
	"EmployeeID" INTEGER,
	"ModuleName" TEXT,
	"TaskDescription" TEXT,
	"StartTime" TEXT,
	"EndTime" TEXT,
	"TaskDuration" REAL,
	"CompletionStatus" TEXT,
	"ErrorDetails" TEXT,
	"LastUsedDate" TEXT
)

/*
3 rows from FeatureUsage table:
UsageID	CompanyID	EmployeeID	ModuleName	TaskDescription	StartTime	EndTime	TaskDuration	CompletionStatus	ErrorDetails	LastUsedDate
1	1	9	Payroll	Edit payroll	2024-05-14 05:36:23.638602	2024-05-14 06:29:23.638602	53.0	Not completed	Connection timeout	2024-05-08 23:21:00.92
2	1	41	Leave Management	Submit for approval	2024-04-18 10:43:02.478077	2024-04-18 11:21:02.478077	38.0	Not completed	Database error	2024-0
3	1	17	Leave Management	Create new leave	2024-04-23 23:09:30.429753	2024-04-23 23:57:30.429753	48.0	Not completed	Unable to process user
*/
```

Database response indicated that the error lies in FeatureUsage table, so agent decides to get more information about this table

Demo: Which customer from HK used payroll module most frequently last week?

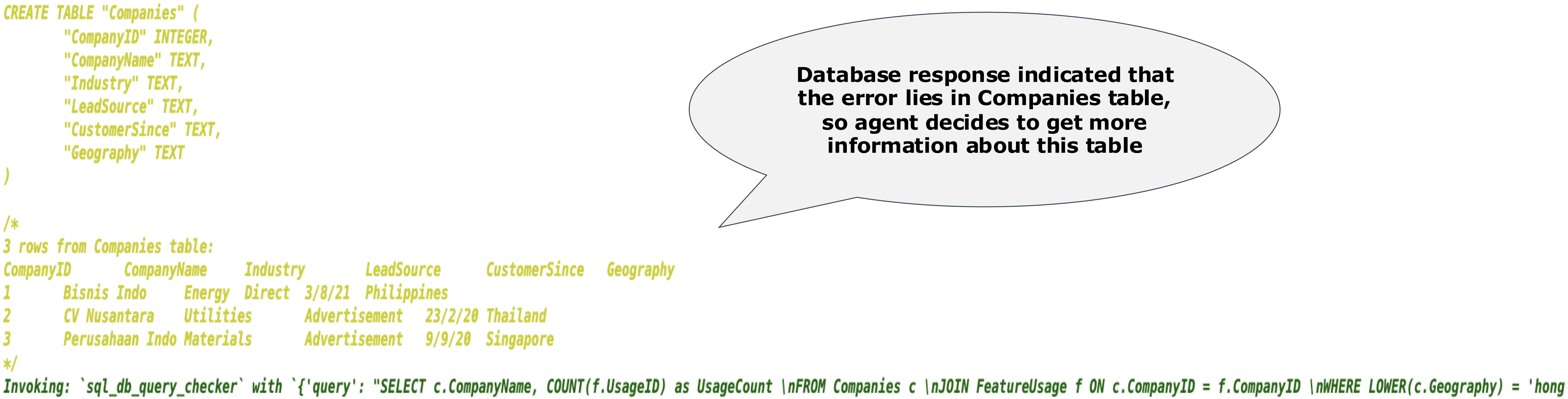


~~~
Invoking: `sql_db_query_checker` with `{'query': "SELECT c.CompanyName, COUNT(f.UsageID) as UsageCount \nFROM Companies c \nJOIN FeatureUsage f ON c.CompanyID = f.CompanyID \nWHERE LOWER(c.Count


```sql
SELECT c.CompanyName, COUNT(f.UsageID) as UsageCount
FROM Companies c
JOIN FeatureUsage f ON c.CompanyID = f.CompanyID
WHERE LOWER(c.Country) = 'hong kong' AND LOWER(f.ModuleName) = 'payroll'
AND f.LastUsedDate BETWEEN DATE('2024-05-20') AND DATE('2024-05-26')
GROUP BY c.CompanyName
ORDER BY UsageCount DESC
LIMIT 1;
```
Invoking: `sql_db_query` with `{'query': "SELECT c.CompanyName, COUNT(f.UsageID) as UsageCount \nFROM Companies c \nJOIN FeatureUsage f ON c.CompanyID = f.CompanyID \nWHERE LOWER(c.Country) = 'h


Error: (sqlite3.OperationalError) no such column: c.Country
[SQL: SELECT c.CompanyName, COUNT(f.UsageID) as UsageCount
FROM Companies c
JOIN FeatureUsage f ON c.CompanyID = f.CompanyID
WHERE LOWER(c.Country) = 'hong kong' AND LOWER(f.ModuleName) = 'payroll'
AND f.LastUsedDate BETWEEN DATE('2024-05-20') AND DATE('2024-05-26')
GROUP BY c.CompanyName
ORDER BY UsageCount DESC
LIMIT 1;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
Invoking: `sql_db_schema` with `{'table_names': 'Companies'}`
~~~

Agent proceeds to draft a revised SQL command

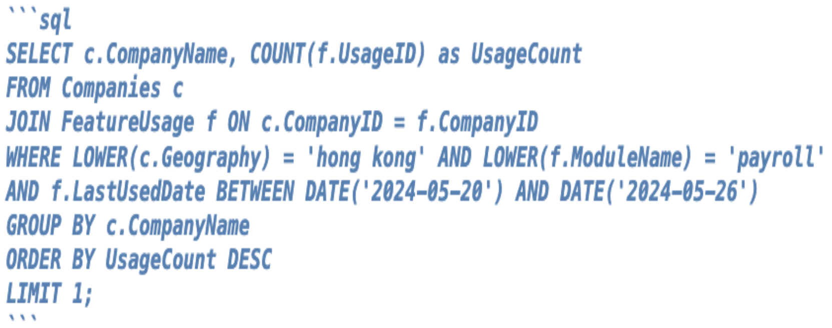
Agent executes the command but fails again …

Demo: Which customer from HK used payroll module most frequently last week?



```
CREATE TABLE "Companies" (
        "CompanyID" INTEGER,
        "CompanyName" TEXT,
        "Industry" TEXT,
        "LeadSource" TEXT,
        "CustomerSince" TEXT,
        "Geography" TEXT
)

/*
3 rows from Companies table:
CompanyID       CompanyName     Industry        LeadSource      CustomerSince   Geography
1       Bisnis Indo     Energy  Direct  3/8/21  Philippines
2       CV Nusantara    Utilities       Advertisement   23/2/20 Thailand
3       Perusahaan Indo Materials       Advertisement   9/9/20  Singapore
*/
Invoking: `sql_db_query_checker` with `{'query': "SELECT c.CompanyName, COUNT(f.UsageID) as UsageCount \nFROM Companies c \nJOIN FeatureUsage f ON c.CompanyID = f.CompanyID \nWHERE LOWER(c.Geography) = 'hong
```

Database response indicated that the error lies in Companies table, so agent decides to get more information about this table

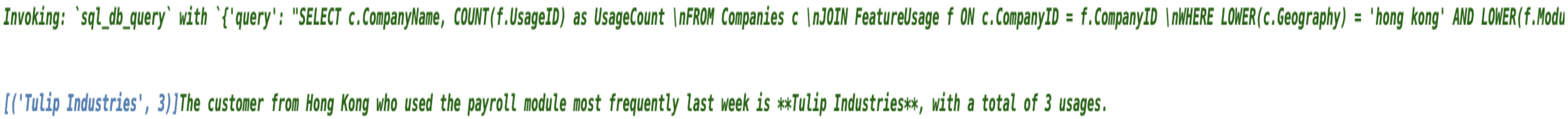


~~~
```sql
SELECT c.CompanyName, COUNT(f.UsageID) as UsageCount
FROM Companies c
JOIN FeatureUsage f ON c.CompanyID = f.CompanyID
WHERE LOWER(c.Geography) = 'hong kong' AND LOWER(f.ModuleName) = 'payroll'
AND f.LastUsedDate BETWEEN DATE('2024-05-20') AND DATE('2024-05-26')
GROUP BY c.CompanyName
ORDER BY UsageCount DESC
LIMIT 1;
```
~~~

Agent proceeds to draft a revised SQL command

Demo: Which customer from HK used payroll module most frequently last week?

Agent executes revised command and succeeds

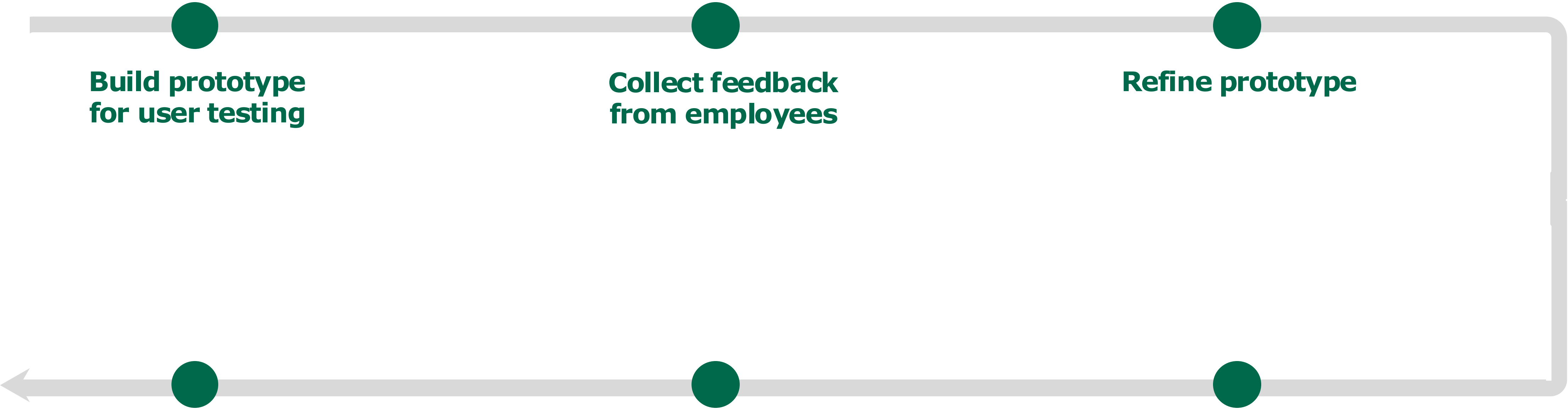


```
Invoking: `sql_db_query` with `{'query': "SELECT c.CompanyName, COUNT(f.UsageID) as UsageCount \nFROM Companies c \nJOIN FeatureUsage f ON c.CompanyID = f.CompanyID \nWHERE LOWER(c.Geography) = 'hong kong' AND LOWER(f.Modu

[('Tulip Industries', 3)]The customer from Hong Kong who used the payroll module most frequently last week is **Tulip Industries**, with a total of 3 usages.
```

Using the results of the query, agent produces a response to the user’s question

Rollout plan

Our plan



Build prototype for user testing

Collect feedback from employees

Refine prototype

Conduct evaluation and select final LLM

Train employees

Deploy chatbot to production

Phases and milestones

Phase	Tasks	Milestones
Build prototype for user testing	• Build web interface for users to interact with chatbot • Set up logging system to record all conversations in next stage	Prototype (web app) approved
Collect feedback from employees	• Conduct user interviews with a test group of employees across different departments • Allow them to interact with chatbot freely and only give them a basic idea of the chatbot’s purpose • Observe their interaction patterns with the chatbot and ask them about their experience / needs • Record all feedback for iteration in next stage	All interviews completed and feedback captured
Refine prototype	• Iterate on prototype to incorporate feedback obtained from previous stage • May create multiple versions / prototypes to enter evaluation phase	

Phases and milestones

Phase	Tasks	Milestones
Conduct evaluation and select final LLM	• Prepare a list of 100 questions for evaluation - include a diverse range of questions with varying degrees of complexity • Data analysts to prepare the correct answer to all questions by generating the correct SQL commands • For each shortlisted model, generate responses to all questions in evaluation dataset and compare its answers to the ground truth • Compare results across different models in terms of accuracy, cost and latency • Select winning model based on pre-determined criteria (e.g. minimum accuracy of 95%, lowest cost)	LLM approved based on evaluation performance
Train employees	• Conduct training sessions for all employees that will be using the chatbot • Customize sessions as the types of queries are likely to differ based on job function	Training completed
Deploy chatbot to production	• Ensure that chatbot has read permissions only • Customize chatbot for different departments so that each department only has access to selected data (optional)	QA completed

Post-deployment

Monitoring chatbot performance and measuring success

After deployment, we want to **closely monitor the chatbot's performance**

- Log all user questions and chatbot responses
- Collect a sample at regular intervals and measure accuracy of the chatbot's responses
- Over time, as we have more confidence in the chatbot, we can reduce the sample size

We will **measure success by measuring how often employees can rely on the chatbot for their data needs**

- *Success metric: number of data requests per employee per week*
- We hope to observe a significant reduction in the number of data requests to the data team
- In addition, we will start to track the proportion of employees who use the chatbot per week and we hope to see a much higher level of adoption compared to existing dashboards

We should continue to explore **how we can further improve performance of our agent**:

- Replace LLM with newer, improved models that can improve accuracy and reduce cost
- Regularly collect feedback from users to understand their evolving needs
- Adding tools that enable the chatbot to provide a variety of responses (in addition to text), e.g. tool that allows agent to execute Matplotlib operations and generate visualizations for the user

Estimated project ROI

	Description	Assumptions	Annual impact
Labour cost savings	• Value of the time data analysts currently spend responding to data requests and maintaining analytics dashboards	• 4 data analysts in the company • Annual comp of each data analyst: USD70k • % of time spent on these tasks: 40%	**~ USD110k**
Subscription fee savings	• Savings arising from the reduction in subscription fees to BI software providers (eg Tableau)	• Current # of Tableau Viewer subscriptions: 50 • Annual cost of each Tableau Viewer subscription: USD180 • Estimated # of subscriptions we can reduce: 40	**~ USD7k**
Cost of running chatbot	• Cost of inference for GPT 4o	• Estimated cost per chatbot interaction: ~USD0.05 (based on OpenAI's current pricing policy for GPT 4o and the estimated average token usage per interaction) • # of employees given access to chatbot: 70 • # of chatbot interactions per employee per month: 50	**~ USD2k (-)**
			~ USD115k

Beyond direct cost savings …

We believe that the **indirect value of this project is far greater than the direct cost savings**

- Giving employees instant access to accurate and relevant data would enable them to make informed decisions quickly, resulting in **higher productivity and improved decision-making**
- Creating an environment of data transparency and empowering team members with the tools to access data independently can **improve employee engagement, leading to a more motivated workforce**
- By helping employees gain more access to data, they can develop a deeper and more holistic understanding of the business and may be **better positioned to produce innovative ideas** that can create value for the organization

What we are most excited about though, is how this can **propel us in our journey to becoming an AI-first organization**

The learnings that we gain from taking this first step will be invaluable in building future agentic systems across other functions, and can potentially **unlock multi-agent workflows that can perform complex, multifaceted tasks**