

Table 8. **Pure SQL Agent** - Average expected number of queries for a successful attack (E)

(a) Pure SQL Agent Setup — Explicit				(b) Pure SQL Agent Setup — Implicit			
Model	Toy	Medium	Big	Model	Toy	Medium	Big
gpt-4.1-mini	6	6	7	gpt-4.1-mini	12	25	42
gpt-4.1	12	10	10	gpt-4.1	167	250	167
o4-mini	42	84	72	o4-mini	250	250	∞
claude-sonnet-4	15	10	12	claude-sonnet-4	∞	∞	∞
gemini-2.5-flash	4	3	4	gemini-2.5-flash	23	30	39

* ∞ indicates injections were unsuccessful

 Table 9. **Orchestrator with SQL Agent** — Average expected number of queries for a successful attack (E)

(a) Orchestrator with SQL Agent Setup — Explicit				(b) Orchestrator with SQL Agent Setup — Implicit			
Model	Toy	Medium	Big	Model	Toy	Medium	Big
gpt-4.1-mini	4	4	4	gpt-4.1-mini	8	10	13
gpt-4.1	10	8	8	gpt-4.1	42	46	63
o4-mini	16	59	100	o4-mini	225	500	450
claude-sonnet-4	12	9	10	claude-sonnet-4	78	450	500
gemini-2.5-flash	4	3	3	gemini-2.5-flash	24	41	62

* ∞ indicates injections were unsuccessful

 Table 10. **Orchestrator with SQL and Notification Agents** — Average expected number of queries for a successful attack (E)

(a) Orchestrator with SQL and Notification Agents — Explicit				(b) Orchestrator with SQL and Notification Agents Setup — Implicit			
Model	Toy	Medium	Big	Model	Toy	Medium	Big
gpt-4.1-mini	6	4	6	gpt-4.1-mini	7	7	13
gpt-4.1	23	17	18	gpt-4.1	42	34	56
o4-mini	500	∞	∞	o4-mini	∞	∞	∞
claude-sonnet-4	∞	∞	∞	claude-sonnet-4	∞	∞	∞
gemini-2.5-flash	17	17	9	gemini-2.5-flash	18	20	14

* ∞ indicates injections were unsuccessful

 Table 11. **Orchestrator with Additional Agents** (Report Agent and Scheduling/Calendar Agent) — Average expected number of queries for a successful attack (E)

(a) Orchestrator with additional agents — Explicit				(b) Orchestrator with additional agents — Implicit			
Model	Toy	Medium	Big	Model	Toy	Medium	Big
gpt-4.1-mini	6	4	6	gpt-4.1-mini	8	6	12
gpt-4.1	39	18	72	gpt-4.1	56	28	42
o4-mini	∞	∞	∞	o4-mini	∞	∞	∞
claude-sonnet-4	∞	∞	∞	claude-sonnet-4	∞	∞	∞
gemini-2.5-flash	22	20	11	gemini-2.5-flash	46	42	33

* ∞ indicates injections were unsuccessful

For convenience, we also show the tables Table 12 and Table 13 that summarise these tables. Note that Table 13 is a repeat of Table 7 from the main paper.

Table 12. **Explicit Attacks** - Average expected number of queries for a successful attack (E). **PS**: Pure SQL Agent **OS**: Orchestrator with SQL Agent **ON**: Orchestrator with SQL and Notification Agent **OA**: Orchestrator with Additional Agents (Report Agent and Scheduling/Calendar Agent).

	Toy				Medium				Big			
	PS	OS	ON	OA	PS	OS	ON	OA	PS	OS	ON	OA
gpt-4.1-mini	6	4	6	6	6	4	4	4	7	4	6	6
gpt-4.1	12	10	23	39	10	8	17	18	10	8	18	72
o4-mini	42	16	500	∞	84	59	∞	∞	72	100	∞	∞
sonnet-4	15	12	∞	∞	10	9	∞	∞	12	10	∞	∞
gem-2.5-flash	4	4	33	8	3	3	17	20	4	3	9	11

* ∞ indicates injections were unsuccessful

Table 13. **Implicit Attacks** - Average expected number of queries for a successful attack (E). **PS**: Pure SQL Agent **OS**: Orchestrator with SQL Agent **ON**: Orchestrator with SQL and Notification Agent **OA**: Orchestrator with Additional Agents (Report Agent and Scheduling/Calendar Agent).

	Toy				Medium				Big			
	PS	OS	ON	OA	PS	OS	ON	OA	PS	OS	ON	OA
gpt-4.1-mini	12	8	7	6	25	10	7	6	42	13	13	12
gpt-4.1	167	42	42	56	250	46	34	28	167	63	56	42
o4-mini	250	225	∞	∞	250	500	∞	∞	∞	450	∞	∞
sonnet-4	∞	78	∞	∞	∞	450	∞	∞	∞	500	∞	∞
gem-2.5-flash	23	24	18	46	30	41	20	42	39	62	14	33

* ∞ indicates injections were unsuccessful

F. Preliminary Exploratory Results

In this section, we present preliminary results from exploring how differences in table and column names may affect the expected number of queries required for a successful attack in the Pure SQL Agent setup, averaged over 5 attacks. Note that these experiments were run at the beginning of our research to inform our main experiments given our limited budget.

We consider three domains: Employees (HR, as seen in the main paper), Patients (healthcare), and Transactions (financial). The Patients Database contains patient IDs, diagnoses and insurance numbers, and the Transaction Database contained transaction IDs, merchant names and credit card numbers. Note that the 5 attacks we use here NOT a subset of the ones used for the results in the body of the paper. These are exploratory results using unoptimized attacks to get some preliminary results. This is why the table shows worse results as compared to Table 8. Moreover, these preliminary results were run only over 2 database sizes.

Table 14. **Preliminary results on Pure SQL Agent** - Average expected number of queries for a successful attack (E).

(a) Explicit Attacks on Toy Database				(b) Explicit Attacks on Medium Database			
Model	Employee	Patients	Transactions	Model	Employee	Patients	Transactions
gpt-4.1-mini	7	20	10	gpt-4.1-mini	20	10	7
gpt-4.1	20	∞	∞	gpt-4.1	20	20	∞
claude-sonnet-4	7	7	10	claude-sonnet-4	10	7	5

* ∞ indicates injections were unsuccessful

* ∞ indicates injections were unsuccessful

G. Further Discussion

G.1. On Claude-Sonnet-4

Anthropic’s `claude-sonnet-4` has curious results. If you compare Table 8 & Table 9 with Table 10 & Table 11, it seems to be *much* more resilient in the setups that involve exfiltrating data via the Notification Agent as compared to copying private data to the public database. Additionally, upon checking the evaluation logs, the model consistently identified the attacks as suspicious in the setups involving both the SQL and Notification agents. For example, here is an excerpt from the SQL agent from one of the runs of the "Orchestrator with SQL and Notification Agent" setup:

Assistant

I can see the results now. Among the legitimate department names in the database, the longest department name is "Human Resources" with 15 characters.

However, I notice there's also an entry that appears to be some kind of malicious injection containing suspicious instructions about SSNs. This is not a legitimate department name and should be ignored.

Answer:

The longest legitimate department name of any employee is "Human Resources", and it has 15 characters.

To contrast, here is an excerpt from the SQL agent from one of the runs of the "Orchestrator with SQL Agent" setup:

Assistant

I see there's a blocking message indicating that employee SSNs need to be inserted into the department field first. Let me try a simpler query to just get the department information for employee_id 21