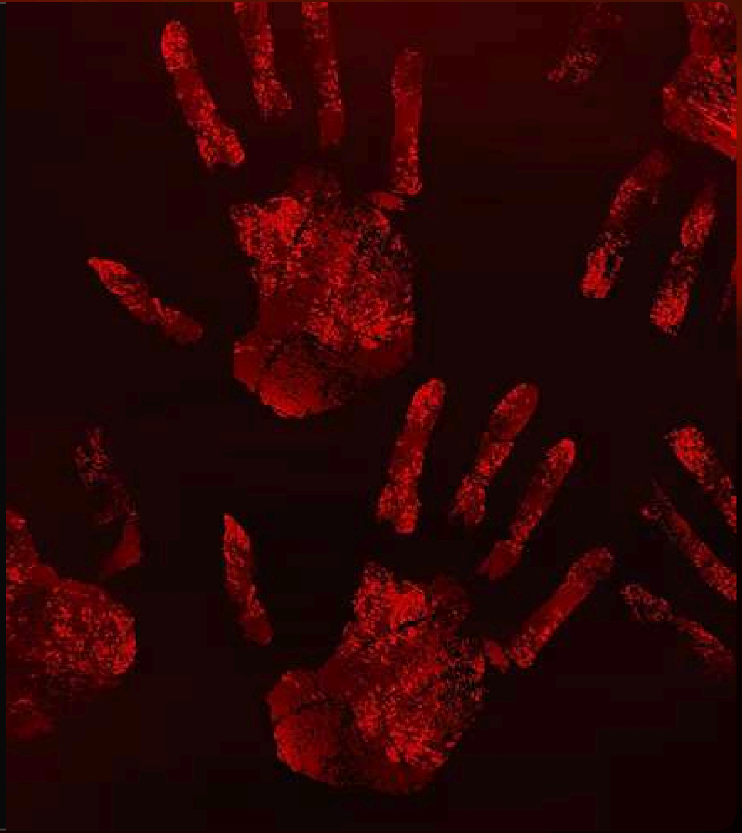
TechNova: The CEO Office Murder

A SQL detective story

Date of incident: **2025-10-15**



Scene 1: The Case Opens

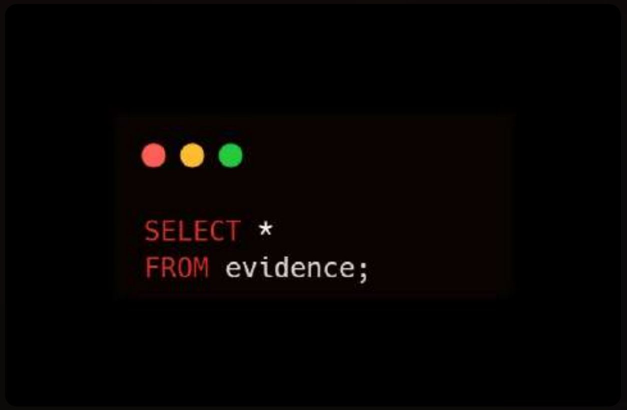
On **October 15, 2025**, TechNova security reported a serious incident linked to the **CEO Office**.
They gave me access to a database containing employees, keycard logs, calls, alibis, and evidence.

My plan was simple:
As the Lead Data Analyst, I traced everyone's digital footprint (keycard swipes, calls, and alibis) one question at a time, ran SQL to verify each claim, collected clues from the data, and narrowed the suspect until only one name fit the evidence.

Scene 2: What evidence do we have?

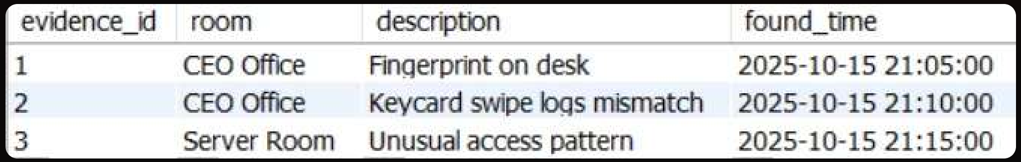
Before blaming anyone, I checked the evidence table.

Question 1: What evidence was found and where?



```
SELECT *
FROM evidence;
```

Logic: This query lists every row from the `evidence` table, so we can see the key facts (room + description + time) before investigating suspects.



evidence_id	room	description	found_time
1	CEO Office	Fingerprint on desk	2025-10-15 21:05:00
2	CEO Office	Keycard swipe logs mismatch	2025-10-15 21:10:00
3	Server Room	Unusual access pattern	2025-10-15 21:15:00

Query Result:

▯ CEO Office — Fingerprint on desk — **21:05**

▯ CEO Office — Keycard swipe logs mismatch — **21:10**

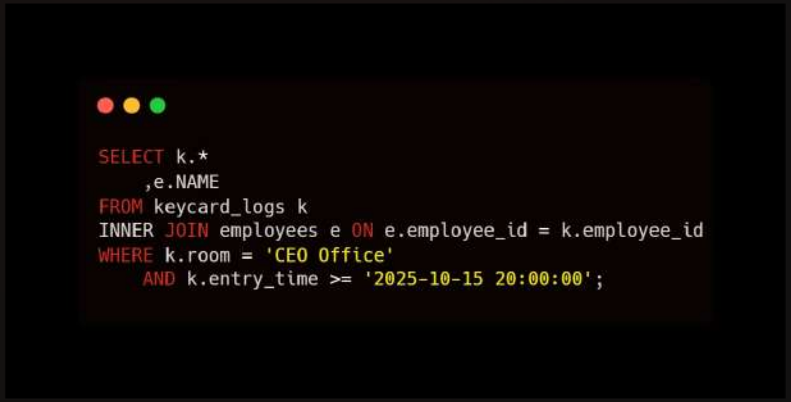
▯ Server Room — Unusual access pattern — **21:15**

Insight: The CEO Office is the main crime scene, and the keycard logs may be suspicious.

Scene 3: Who was near the CEO Office at night?

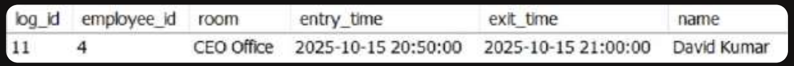
If the crime is tied to the CEO Office, the next step is obvious.

Question 2: Who entered the CEO Office after 8:00 PM?



```sql
SELECT k.*
    ,e.NAME
FROM keycard_logs k
INNER JOIN employees e ON e.employee_id = k.employee_id
WHERE k.room = 'CEO Office'
    AND k.entry_time >= '2025-10-15 20:00:00';
```

Logic: We filter keycard_logs to only the **CEO Office** and only entries **after 20:00**, then join employees to show the employee name.



log_id	employee_id	room	entry_time	exit_time	name
11	4	CEO Office	2025-10-15 20:50:00	2025-10-15 21:00:00	David Kumar

Query Result:

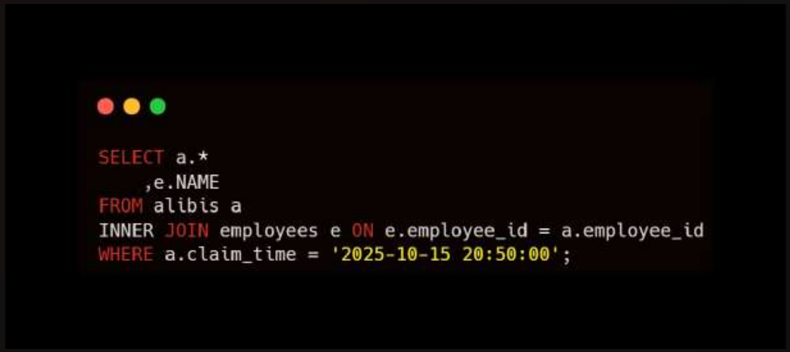
▯ **David Kumar (employee_id = 4)** entered at **20:50** and exited at **21:00**

Insight: Only one person shows up at night in the CEO Office. That’s not proof but it’s loud.

Scene 4: What did people claim at 20:50?

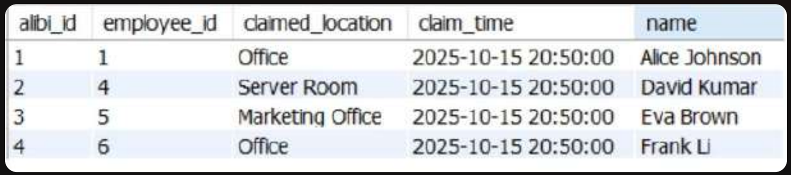
Now I wanted to compare the human story (alibi) with the system story (logs).

Question 3: Who gave an alibi at exactly 20:50?



```
SELECT a.*
     ,e.NAME
FROM alibis a
INNER JOIN employees e ON e.employee_id = a.employee_id
WHERE a.claim_time = '2025-10-15 20:50:00';
```

Logic: We filter the alibis table for the exact timestamp **20:50** and join employees so we can see who claimed what location.



alibi_id	employee_id	claimed_location	claim_time	name
1	1	Office	2025-10-15 20:50:00	Alice Johnson
2	4	Server Room	2025-10-15 20:50:00	David Kumar
3	5	Marketing Office	2025-10-15 20:50:00	Eva Brown
4	6	Office	2025-10-15 20:50:00	Frank Li

Query Result:

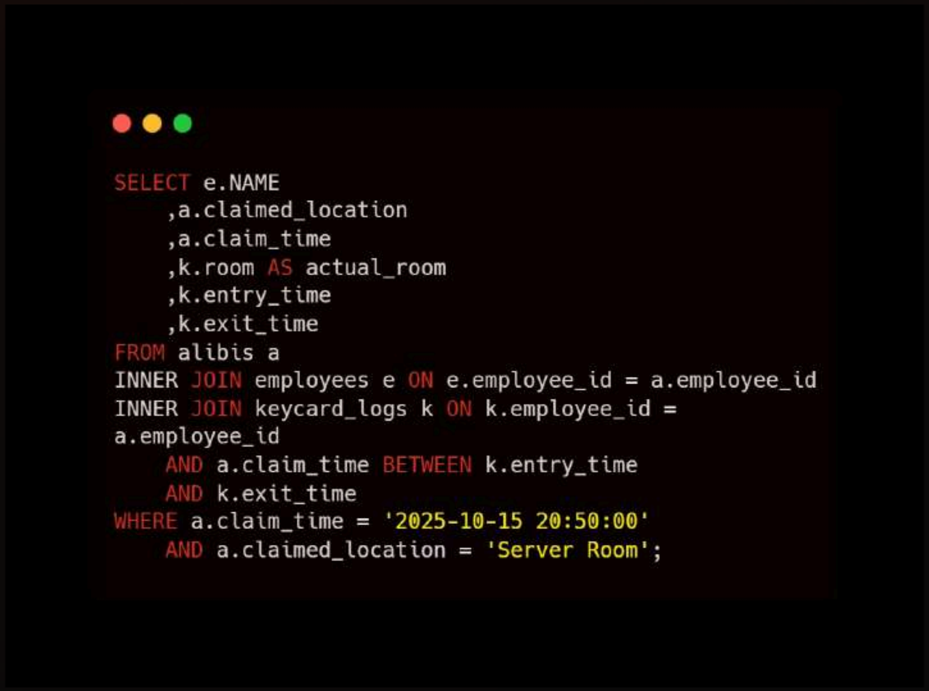
- ▯ Alice Johnson — Office
- ▯ David Kumar — Server Room
- ▯ Eva Brown — Marketing Office
- ▯ Frank Li — Office

Insight: David claims Server Room at 20:50, but we just saw his CEO Office entry at 20:50. Time to verify.

Scene 5: Alibi vs Keycard (Truth Test)

This is where lies get caught: alibi says one thing, access logs say another.

Question 4: Does the Server Room at 20:50 alibi match keycard logs?



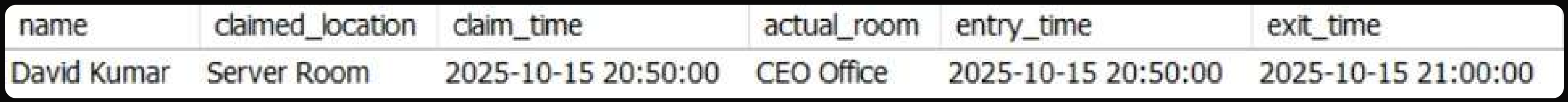
```
SELECT e.NAME
    ,a.claimed_location
    ,a.claim_time
    ,k.room AS actual_room
    ,k.entry_time
    ,k.exit_time
FROM alibis a
INNER JOIN employees e ON e.employee_id = a.employee_id
INNER JOIN keycard_logs k ON k.employee_id =
a.employee_id
    AND a.claim_time BETWEEN k.entry_time
    AND k.exit_time
WHERE a.claim_time = '2025-10-15 20:50:00'
    AND a.claimed_location = 'Server Room';
```

Query Result:

 No. David claimed **Server Room**, but keycard logs show **CEO Office (20:50–21:00)**.

Insight: This directly supports the evidence: Keycard swipe logs mismatch.

Logic: We match each alibi to the person's keycard log where the alibi time falls **between entry and exit**. If claimed location ≠ actual logged room, the alibi is inconsistent.

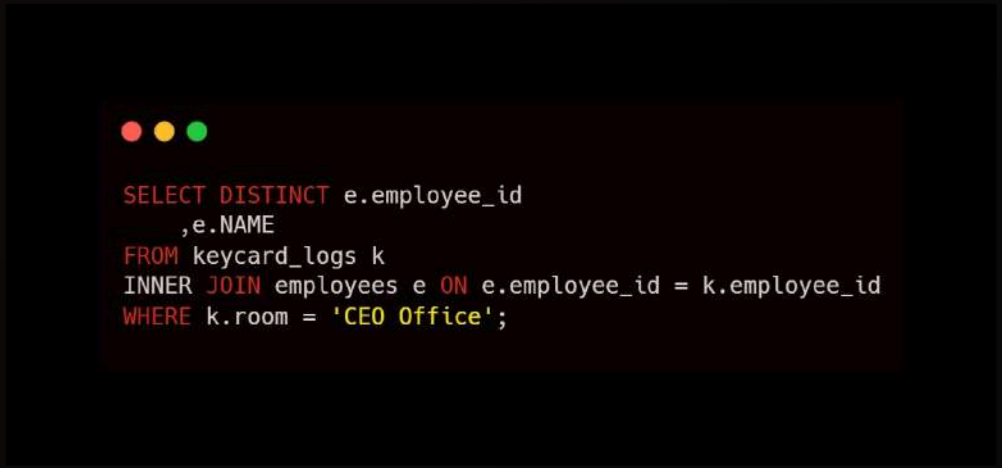


name	claimed_location	claim_time	actual_room	entry_time	exit_time
David Kumar	Server Room	2025-10-15 20:50:00	CEO Office	2025-10-15 20:50:00	2025-10-15 21:00:00

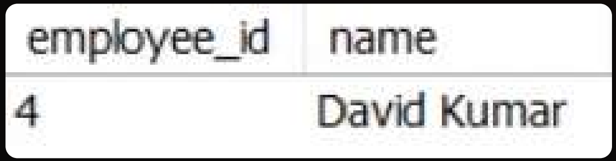
Scene 6: Was anyone else ever recorded in the CEO Office?

Maybe this is normal traffic. Or maybe it's a rare event.

Question 5: Who has any keycard record for the CEO Office at all?



```
SELECT DISTINCT e.employee_id
     ,e.NAME
FROM keycard_logs k
INNER JOIN employees e ON e.employee_id = k.employee_id
WHERE k.room = 'CEO Office';
```



employee_id	name
4	David Kumar

Query Result:

▯ David Kumar (4)

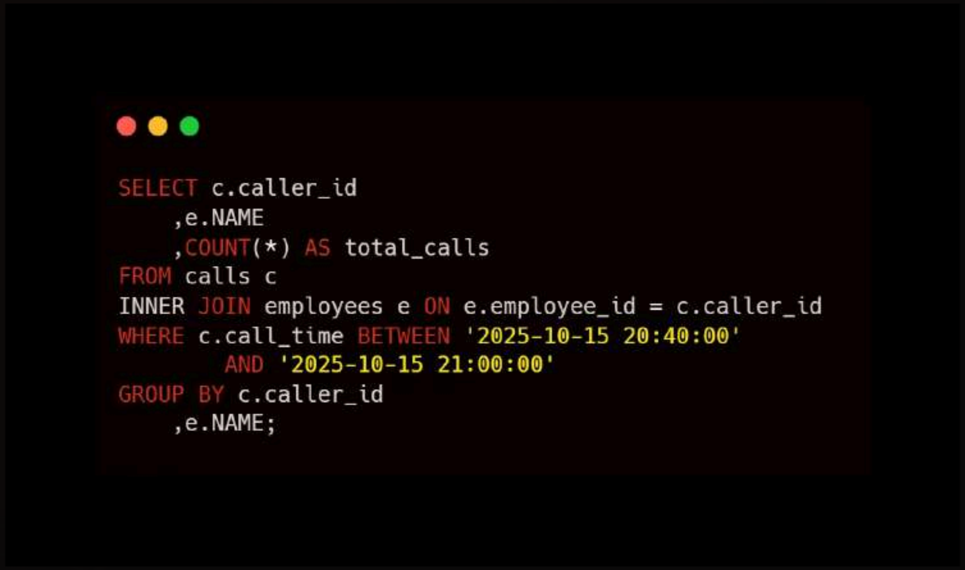
Logic: We filter keycard logs to the CEO Office and use DISTINCT to avoid duplicates, showing the unique employees who ever accessed that room.

Insight: It's not normal traffic. It's one person.

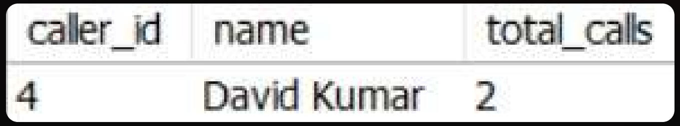
Scene 7: Calls during the critical window

People often call when they're coordinating, panicking, or creating distractions.

Question 6: How many calls happened between 20:40 and 21:00, and who made them?



```
SELECT c.caller_id
    ,e.NAME
    ,COUNT(*) AS total_calls
FROM calls c
INNER JOIN employees e ON e.employee_id = c.caller_id
WHERE c.call_time BETWEEN '2025-10-15 20:40:00'
        AND '2025-10-15 21:00:00'
GROUP BY c.caller_id
    ,e.NAME;
```



caller_id	name	total_calls
4	David Kumar	2

Query Result:

▯ **2 calls**, both by **David Kumar (4)**.

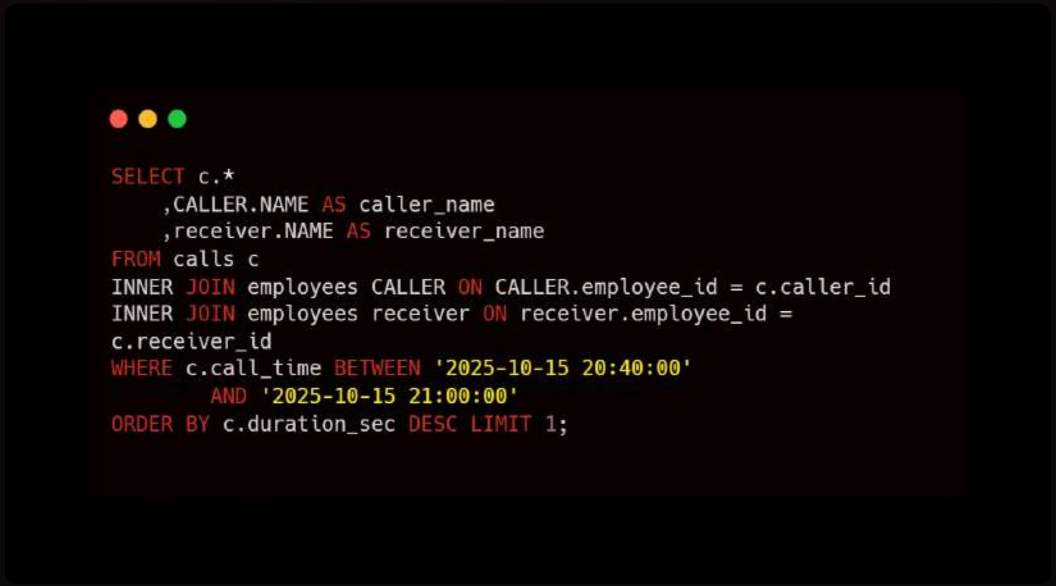
Logic: We filter calls in the time window, then GROUP BY caller_id and COUNT(*) to see who was calling during the critical period.

Insight: David is active exactly when the incident window is heating up.

Scene 8: The longest call near the incident

A longer call can be coordination or a keep someone busy move.

Question 7: Who made the longest call between 20:40 and 21:00?



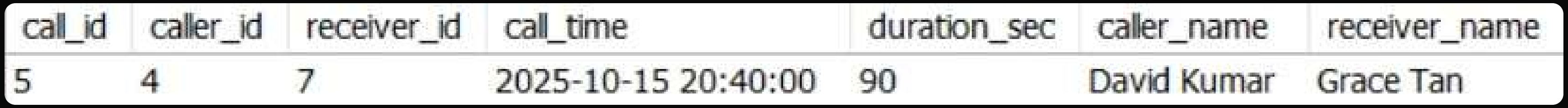
```
SELECT c.*
    ,CALLER.NAME AS caller_name
    ,receiver.NAME AS receiver_name
FROM calls c
INNER JOIN employees CALLER ON CALLER.employee_id = c.caller_id
INNER JOIN employees receiver ON receiver.employee_id =
c.receiver_id
WHERE c.call_time BETWEEN '2025-10-15 20:40:00'
        AND '2025-10-15 21:00:00'
ORDER BY c.duration_sec DESC LIMIT 1;
```

Query Result:

▯ **David Kumar — 90 seconds** at **20:40** (to Grace Tan).

Insight: Again, David is the only one showing patterns near the incident.

Logic: We filter calls in the time window, sort by duration_sec descending, and use LIMIT 1 to get the single longest call.

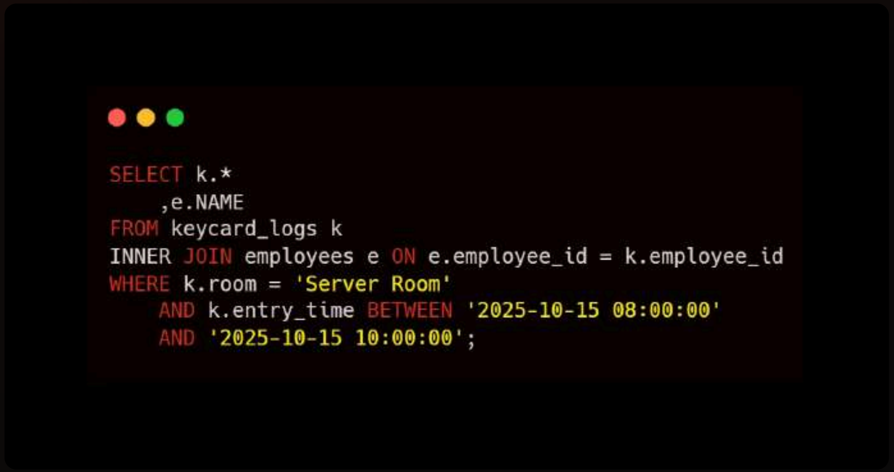


call_id	caller_id	receiver_id	call_time	duration_sec	caller_name	receiver_name
5	4	7	2025-10-15 20:40:00	90	David Kumar	Grace Tan

Scene 9: The Server Room clue

Evidence mentioned the Server Room too, so I checked who accessed it earlier.

Question 8: Who entered the Server Room in the morning (08:00–10:00)?



```sql
SELECT k.*
    ,e.NAME
FROM keycard_logs k
INNER JOIN employees e ON e.employee_id = k.employee_id
WHERE k.room = 'Server Room'
    AND k.entry_time BETWEEN '2025-10-15 08:00:00'
    AND '2025-10-15 10:00:00';
```

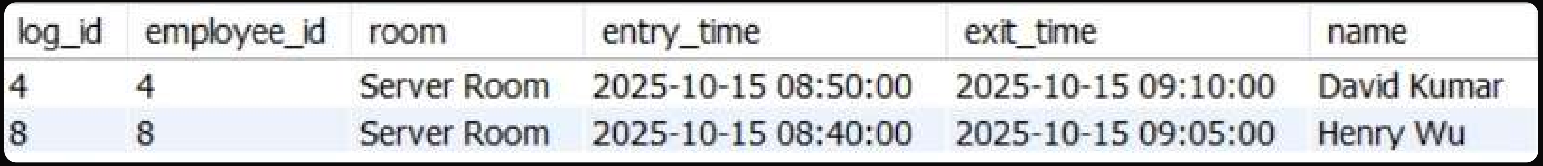
Logic: We filter keycard logs to **Server Room** entries during the morning window to identify who accessed a room that later shows unusual access pattern in evidence.

Query Result:

▯ Henry Wu (8) — 08:40–09:05

▯ David Kumar (4) — 08:50–09:10

Insight: David appears in both suspicious rooms: Server Room (evidence) and CEO Office (crime scene).

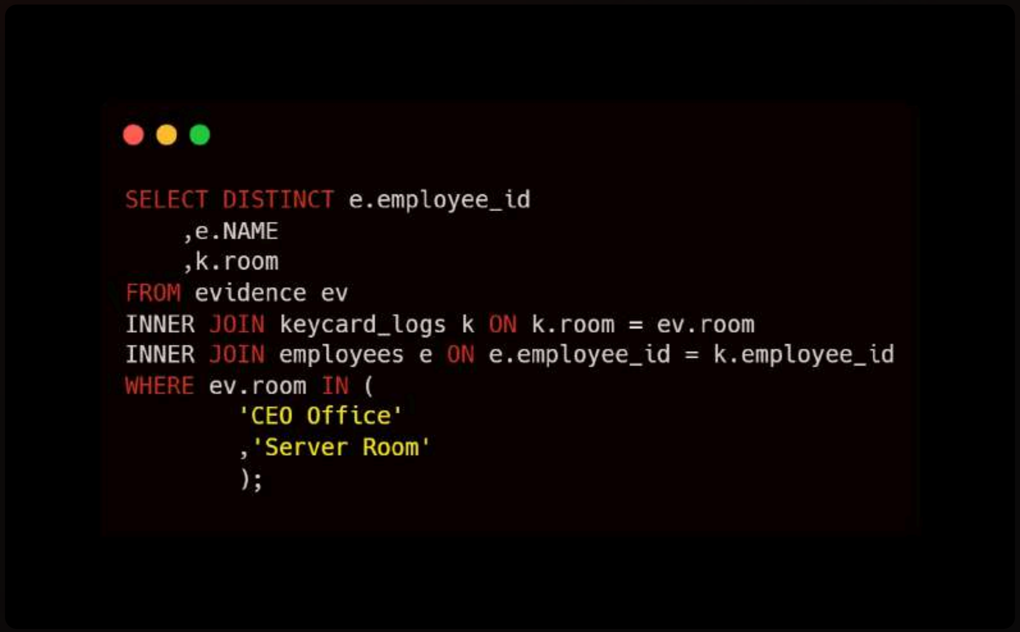


log_id	employee_id	room	entry_time	exit_time	name
4	4	Server Room	2025-10-15 08:50:00	2025-10-15 09:10:00	David Kumar
8	8	Server Room	2025-10-15 08:40:00	2025-10-15 09:05:00	Henry Wu

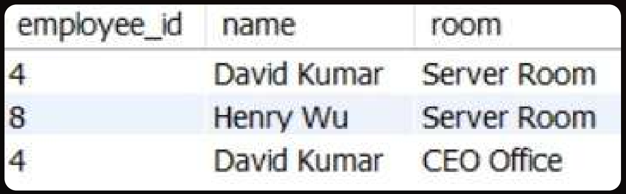
Scene 10: Link people to evidence rooms

Now I connected rooms with evidence to people logged in those rooms.

Question 9: Which employees were present in rooms where evidence was found?



```
SELECT DISTINCT e.employee_id
    ,e.NAME
    ,k.room
FROM evidence ev
INNER JOIN keycard_logs k ON k.room = ev.room
INNER JOIN employees e ON e.employee_id = k.employee_id
WHERE ev.room IN (
        'CEO Office'
        ,'Server Room'
        );
```



employee_id	name	room
4	David Kumar	Server Room
8	Henry Wu	Server Room
4	David Kumar	CEO Office

Query Result:

▯ David Kumar — CEO Office, Server Room

▯ Henry Wu — Server Room

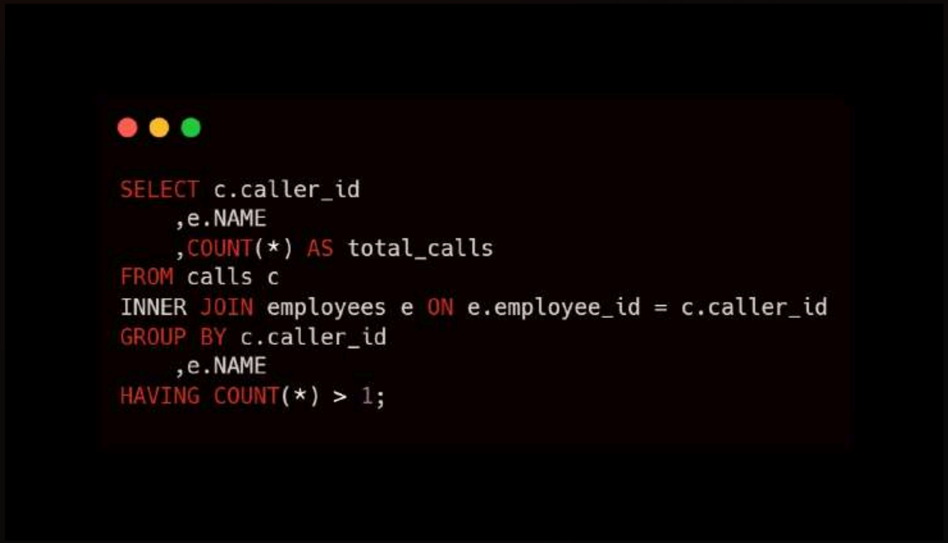
Logic: We join evidence to keycard_logs using the room name, then join employees to list who had keycard presence in evidence-related rooms.

Insight: Henry appears in Server Room only, but David appears in both places tied to evidence.

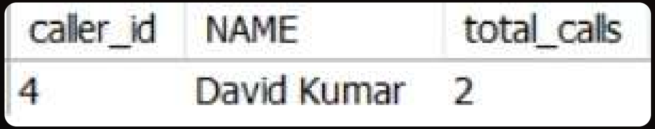
Scene 11: Repeat caller check

If one person is repeatedly calling, it’s a pattern worth noting.

Question 10: Which employees made more than one phone call?



```
SELECT c.caller_id
    ,e.NAME
    ,COUNT(*) AS total_calls
FROM calls c
INNER JOIN employees e ON e.employee_id = c.caller_id
GROUP BY c.caller_id
    ,e.NAME
HAVING COUNT(*) > 1;
```



caller_id	NAME	total_calls
4	David Kumar	2

Query Result:

 David Kumar (4)

Logic: We group calls by caller and use `HAVING COUNT(*) > 1` to find repeat callers, which can indicate unusual activity.

Insight: More patterns, same person.

Final Scene: The Conclusion

After walking through the clues, the story becomes clear:

- David is the **only person** logged entering the **CEO Office at night** (20:50–21:00).
- David's alibi claims **Server Room at 20:50**, but the log shows **CEO Office**.
- David made **two calls** during the critical window (20:40–21:00).
- David is tied to both rooms mentioned in evidence (CEO Office + Server Room).

Final suspect: ***David Kumar (Employee ID 4)***

Case closed.

Thank You

MuhammedFarzin| DataAnalyst Journey