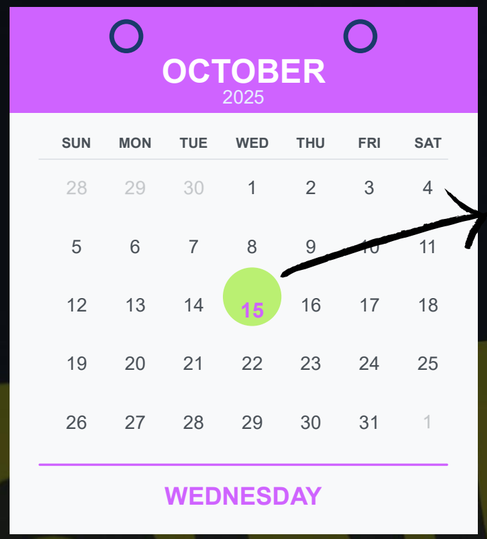
SQL Murder Mystery:
Who killed the CEO?

Case solved by Rishabh Bahuguna

At the crime scene

The CEO of TechNova Inc. was found dead in their office.



09:00 PM

21:00 Hrs. as per our database.

Who am **I** and what is **my mission**?

I am a lead data analyst at TechNova Inc., and my mission is to find out:

a) who the killer is

b) where and when the crime took place, and

c) how it happened

	employee_id [PK] integer	name character varying (50)	department character varying (50)	role character varying (50)
1	1	Alice Johnson	Engineering	Software Engineer
2	2	Bob Smith	HR	HR Manager
3	3	Clara Lee	Finance	Accountant
4	4	David Kumar	Engineering	DevOps Engineer
5	5	Eva Brown	Marketing	Marketing Lead
6	6	Frank Li	Engineering	QA Engineer
7	7	Grace Tan	Finance	CFO
8	8	Henry Wu	Engineering	CTO
9	9	Isla Patel	Support	Customer Support
10	10	Jack Chen	HR	Recruiter

employees table

Tables: where our clues are hidden.

This will help us explore the evidence concealed within the data.

✅ **Concept check: Here is the overview of table basics in Data Definition Language.**

	log_id [PK] integer	employee_id integer	room character varying (50)	entry_time timestamp without time zone	exit_time timestamp without time zone
1	1	1	Office	2025-10-15 08:00:00	2025-10-15 12:00:00
2	2	2	HR Office	2025-10-15 08:30:00	2025-10-15 17:00:00
3	3	3	Finance Office	2025-10-15 08:45:00	2025-10-15 12:30:00
4	4	4	Server Room	2025-10-15 08:50:00	2025-10-15 09:10:00
5	5	5	Marketing Office	2025-10-15 09:00:00	2025-10-15 17:30:00
6	6	6	Office	2025-10-15 08:30:00	2025-10-15 12:30:00
7	7	7	Finance Office	2025-10-15 08:00:00	2025-10-15 18:00:00
8	8	8	Server Room	2025-10-15 08:40:00	2025-10-15 09:05:00
9	9	9	Support Office	2025-10-15 08:30:00	2025-10-15 16:30:00
10	10	10	HR Office	2025-10-15 09:00:00	2025-10-15 17:00:00
11	11	4	CEO Office	2025-10-15 20:50:00	2025-10-15 21:00:00

keycards_log table

✅ Concept check: CREATE TABLE is a Data Definition Language (DDL) command used to define the structure of a new table within a database.

	call_id [PK] integer	caller_id integer	receiver_id integer	call_time timestamp without time zone	duration_sec integer
1	1	4	1	2025-10-15 20:55:00	45
2	2	5	1	2025-10-15 19:30:00	120
3	3	3	7	2025-10-15 14:00:00	60
4	4	2	10	2025-10-15 16:30:00	30
5	5	4	7	2025-10-15 20:40:00	90

calls table

✅ Concept check: ALTER TABLE is a DDL command used to modify the structure of a table within a database.

	alibi_id [PK] integer	employee_id integer	claimed_location character varying (50)	claim_time timestamp without time zone
	1	1	Office	2025-10-15 20:50:00
	2	4	Server Room	2025-10-15 20:50:00
	3	5	Marketing Office	2025-10-15 20:50:00
	4	6	Office	2025-10-15 20:50:00

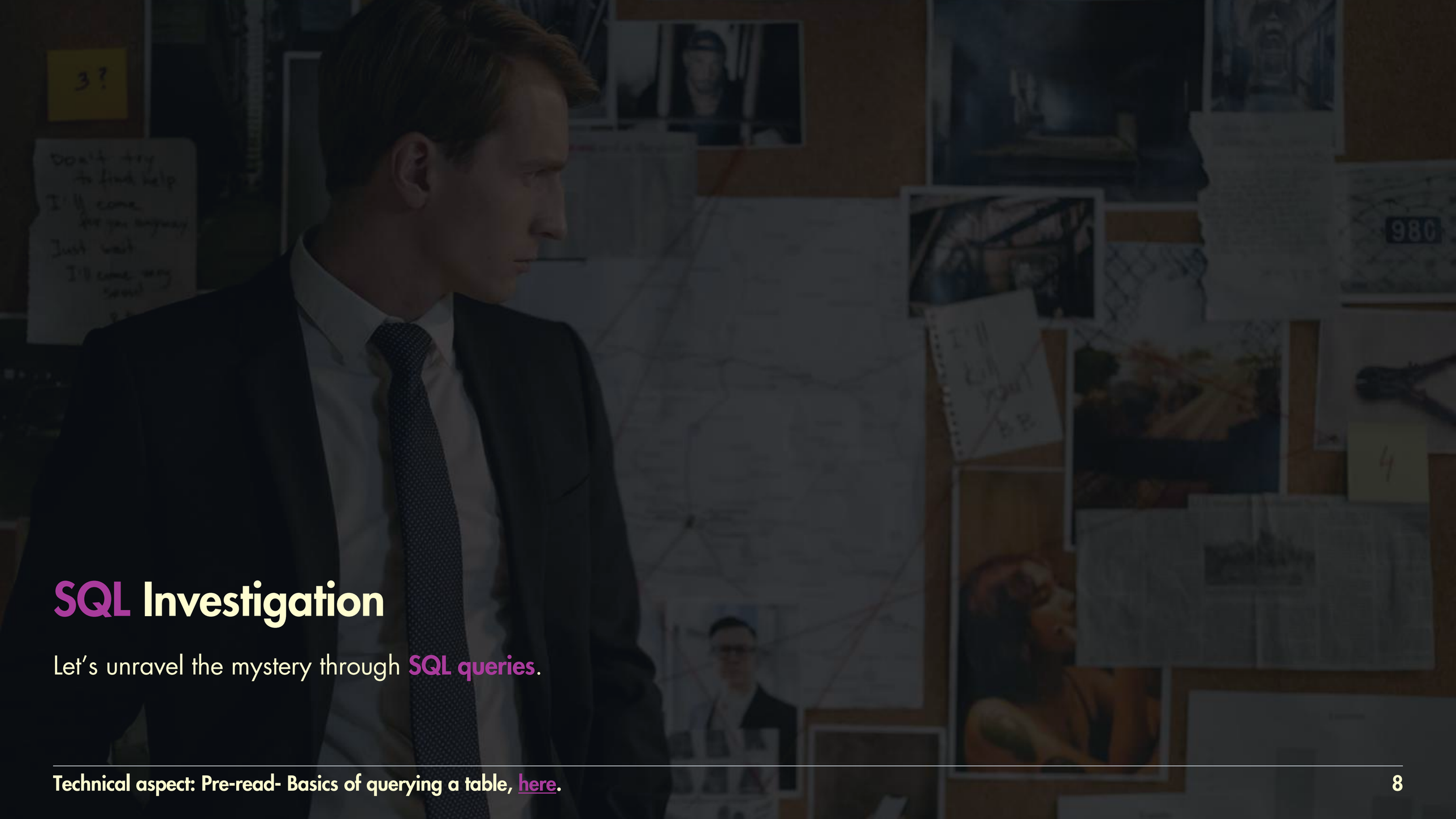
alibis table

✅ Concept check: DROP TABLE is a DDL command used to delete the table and its components in a database permanently.

evidence_id [PK] integer	room character varying (50)	description character varying (255)	found_time timestamp without time zo
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

evidence table

✅ Concept check: TRUNCATE TABLE is a DDL command used to remove all the rows from the table while preserving the table's structure within a database.

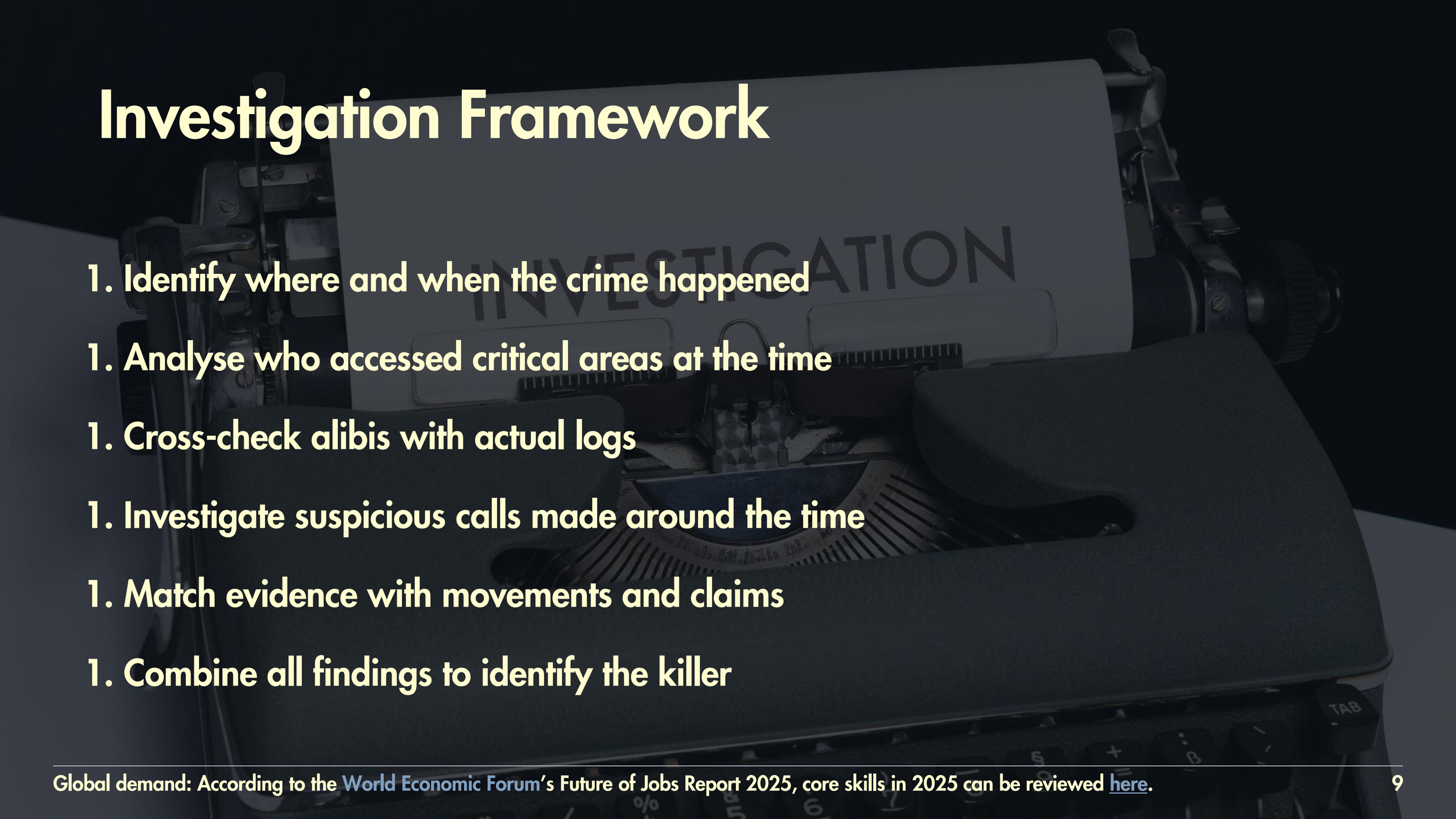


SQL Investigation

Let's unravel the mystery through **SQL queries**.

Technical aspect: Pre-read- Basics of querying a table, here.

Investigation Framework



1. Identify where and when the crime happened
1. Analyse who accessed critical areas at the time
1. Cross-check alibis with actual logs
1. Investigate suspicious calls made around the time
1. Match evidence with movements and claims
1. Combine all findings to identify the killer

Global demand: According to the World Economic Forum's Future of Jobs Report 2025, core skills in 2025 can be reviewed here.

1.Identify where and when the crime happened

SQL Query:

```
SELECT room AS crime_scene, found_time AS time_discovered, description
FROM evidence
WHERE room = 'CEO Office';
```

Investigation result:

crime_scene character varying (50)	time_discovered timestamp without time zone	description character varying (255)
CEO Office	2025-10-15 21:05:00	Fingerprint on desk
CEO Office	2025-10-15 21:10:00	Keycard swipe logs mismatch

Crime scene findings:

The evidence confirms the crime took place inside the **CEO Office**, with traces appearing **between 21:05 and 21:10 Hrs.**

1. Analyse who accessed critical areas at the time

SQL Query:

```
SELECT e.employee_id, e.name, kl.log_id, kl.room, kl.entry_time, kl.exit_time
FROM employees e
JOIN keycard_logs kl ON e.employee_id = kl.employee_id
WHERE room = 'CEO Office' AND entry_time BETWEEN '2025-10-15 20:30:00' AND '2025-10-15 21:10:00';
```

Investigation result:

employee_id integer	name character varying (50)	log_id integer	room character varying (50)	entry_time timestamp without time zone	exit_time timestamp without time zone
4	David Kumar	11	CEO Office	2025-10-15 20:50:00	2025-10-15 21:00:00

Crime scene findings:

The access logs show that **only one person** entered the **CEO's Office** within the critical window, narrowing the investigation to **a single potential suspect.**

Concept check: Filtering with the **BETWEEN operator** checks whether a value falls under a **specified range**.

1.Cross-check alibis with actual log

SQL Query:

```
SELECT a.*, kl.log_id, kl.room, kl.entry_time, kl.exit_time
FROM alibis a
LEFT JOIN keycard_logs kl
ON a.employee_id = kl.employee_id
AND a.claim_time BETWEEN kl.entry_time AND kl.exit_time
ORDER BY alibi_id;
```

Investigation result:

alibi_id integer	employee_id integer	claimed_location character varying (50)	claim_time timestamp without time zone	log_id integer	room character varying (50)	entry_time timestamp without time zone	exit_time timestamp without time zone
1	1	Office	2025-10-15 20:50:00	[null]	[null]	[null]	[null]
2	4	Server Room	2025-10-15 20:50:00	11	CEO Office	2025-10-15 20:50:00	2025-10-15 21:00:00
3	5	Marketing Office	2025-10-15 20:50:00	[null]	[null]	[null]	[null]
4	6	Office	2025-10-15 20:50:00	[null]	[null]	[null]	[null]

Crime scene findings:

Only one alibi matches with an actual keycard entry. The employee who claimed to be in the CEO's Office at 20:50 Hrs. is the same one whose swipe log confirms access at that time.

1.Investigate suspicious calls made around the time

SQL Query:

```
SELECT c.caller_id, e1.name AS caller_name, c.receiver_id, e2.name AS receiver_name, c.call_time, c.duration_sec
FROM employees e1
LEFT JOIN calls c
ON e1.employee_id = c.caller_id
LEFT JOIN employees e2
ON e2.employee_id = c.receiver_id
WHERE c.call_time BETWEEN '2025-10-15 20:50:00' AND '2025-10-15 21:00:00';
```

Investigation result:

caller_id integer	caller_name character varying (50)	receiver_id integer	receiver_name character varying (50)	call_time timestamp without time zone	duration_sec integer
4	David Kumar	1	Alice Johnson	2025-10-15 20:55:00	45

Crime scene findings:

Only one call took place within the time window, indicating that **David Kumar** contacted **Alice Johnson** at **20:55 Hrs.**, making it **a key interaction** to note during the incident.

✅ **Concept check: LEFT JOIN operation** retains all rows from **the left table** and adds matching rows from the right, displaying **nulls** when no match exists.

1.Match evidence with movements and claims

SQL Query:

```sql
SELECT
    e.*,
    kl.employee_id,
    e1.name,
    kl.log_id,
    kl.entry_time,
    kl.exit_time,
    a.claim_time,
    a.claimed_location
FROM evidence e
LEFT JOIN keycard_logs kl
    ON e.room = kl.room
LEFT JOIN employees e1
    ON kl.employee_id = e1.employee_id
LEFT JOIN alibis a
    ON kl.employee_id = a.employee_id
WHERE e.found_time BETWEEN kl.entry_time
    AND (kl.exit_time + interval '15 minutes');
```

Investigation result:

Note: The complete result set contains 11 columns.

evidence_id integer	room character varying (50)	description character varying (255)	found_time timestamp without time zone	employee_id integer	name character varying (50)	log_id integer
1	CEO Office	Fingerprint on desk	2025-10-15 21:05:00	4	David Kumar	11
2	CEO Office	Keycard swipe logs mismatch	2025-10-15 21:10:00	4	David Kumar	11

log_id integer	entry_time timestamp without time zone	exit_time timestamp without time zone	claim_time timestamp without time zone	claimed_location character varying (50)
11	2025-10-15 20:50:00	2025-10-15 21:00:00	2025-10-15 20:50:00	Server Room
11	2025-10-15 20:50:00	2025-10-15 21:00:00	2025-10-15 20:50:00	Server Room

Crime scene findings:

All evidence in the **CEO's Office**, including the **fingerprint and keycard mismatch**, aligns with **David Kumar**'s movements and claimed location, linking him directly to the scene during the incident.

✅ **Concept check: LEFT JOIN operation** retains all rows from **the left table** and adds matching rows from the right, displaying **nulls** when no match exists.

1.Combine all findings to identify the killer.

SQL Query:

cte_1

```
WITH cte_key AS (
SELECT
    e.employee_id,
    e.name,
    'keycard_logs' AS match_found_in
FROM employees e
LEFT JOIN keycard_logs kl
ON e.employee_id = kl.employee_id
WHERE kl.room = 'CEO Office'
),
```

cte_2

```
cte_calls AS (
SELECT
    e.employee_id,
    e.name,
    'calls' AS match_found_in
FROM employees e
LEFT JOIN calls c
ON e.employee_id = c.caller_id
WHERE c.call_time BETWEEN '2025-10-15 20:30:00' AND '2025-10-15 21:10:00'
),
```

cte_3

```
cte_alibis AS (
SELECT
    e.employee_id,
    e.name,
    'alibis' AS match_found_in
FROM employees e
LEFT JOIN alibis a
ON e.employee_id = a.employee_id
LEFT JOIN keycard_logs kl
ON a.employee_id = kl.employee_id
WHERE a.claim_time BETWEEN '2025-10-15 20:30:00' AND '2025-10-15 21:10:00'
AND kl.room <> a.claimed_location
),
```

cte_4

```
cte_evidence AS (
SELECT
    e.employee_id,
    e.name,
    'evidence' AS match_found_in
FROM employees e
LEFT JOIN keycard_logs kl
ON e.employee_id = kl.employee_id
LEFT JOIN evidence ev
ON kl.room = ev.room
WHERE ev.found_time BETWEEN kl.entry_time AND (kl.exit_time + interval '15 minutes')
)
```

✅ Concept check: Common Table Expressions (CTEs) create temporary named result sets that only exist during the execution of a single query.

Behind the alibi: a killer exposed

SQL Query:

Note: Part of a larger CTE chain. Refer to the previous slide for the WITH clause

```sql
SELECT name AS killer FROM cte_key
UNION
SELECT name AS killer FROM cte_calls
UNION
SELECT name AS killer FROM cte_alibis
UNION
SELECT name AS killer FROM cte_evidence;
```

Investigation result:

killer character varying (50)
David Kumar

✅ Concept check: UNION operator combines the result sets of two or more SELECT statements into a single result set, by default removing duplicate rows. 16

Case closed

We've found the killer using our investigation with SQL

Thank you for your attention

Special thanks to:

and

INDIAN DATA CLUB

DPDzero

For this investigation, I, Rishabh Bahuguna, took on the role of lead data analyst at TechNova Inc. and used my SQL and detective skills to work through the case.

Want to know more about this analyst (with honed detective skills)?
Reach out with your case here: