Day 2 — Advanced SQL Theory (20 Minutes)

1. Window Functions
Window functions perform calculations across related rows without reducing the number of rows.
Common window functions:
• ROW_NUMBER() — unique ranking
• RANK() — ranking with gaps on ties
• DENSE_RANK() — ranking without gaps
• SUM() OVER() — running totals
• AVG() OVER() — moving averages

Example:
```
SELECT customer_id, amount,
ROW_NUMBER() OVER (ORDER BY amount DESC) AS rn
FROM orders;
```

2. Differences Between ROW_NUMBER, RANK, and DENSE_RANK
Values: 100, 100, 90
• ROW_NUMBER → 1, 2, 3
• RANK → 1, 1, 3
• DENSE_RANK → 1, 1, 2

3. Correlated Subqueries
A subquery that depends on the outer query and runs once per row.
Example:
```
SELECT e1.name
FROM employees e1
WHERE salary > (
SELECT AVG(salary)
FROM employees e2
WHERE e1.department = e2.department
);
```

4. Non-Correlated Subqueries
A subquery that runs independently of the outer query.
Example:
```
SELECT name FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

5. Indexes
Indexes speed up searches by creating a lookup structure.
Use indexes on columns often used in:
• WHERE
• JOIN
• ORDER BY

Avoid indexing:
• Low-selectivity columns (e.g., gender)
• Small tables
• Frequently updated columns

Example:
```
CREATE INDEX idx_customer_id ON orders(customer_id);
```

6. CTE (WITH clause)

Used to simplify complex queries.
Example:

```
WITH total_sales AS (
SELECT customer_id, SUM(amount) AS total
FROM orders
GROUP BY customer_id
)
SELECT * FROM total_sales WHERE total > 10000;
```

7. Nth Highest Salary Pattern
Use ROW_NUMBER or DENSE_RANK.
Example for 3rd highest:

```
SELECT salary FROM (
SELECT salary,
ROW_NUMBER() OVER (ORDER BY salary DESC) AS rn
FROM employees
) t
WHERE rn = 3;
```