MTHREE SQL Training - Day 2

📌 Permissions using Binary Operations

In SQL, permissions are managed using **binary operations** to evaluate multiple permission states. Permissions are represented as bits, and operations like **AND, OR, NOT, XOR** are used to manipulate them.

🔹 Common Use Cases:
1. **Permission Representation:**
   - `001` → Read
   - `010` → Write
   - `100` → Execute

2. **Granting Permissions (OR `|`):**
   ```sql
   -- Grant Read (001) and Write (010):
   001 | 010 = 011 (Read + Write)
   ```

3. **Revoking Permissions (AND `&` with NOT `~`):**
   ```sql
   -- Revoke Write (010) from Read+Write (011):
   011 & ~010 = 001 (Only Read)
   ```

4. **Checking Permissions (AND `&`):**
   ```sql
   -- Check if Write (010) is granted in 011:
   011 & 010 = 010 (Write exists)
   ```

🔹 SQL Example:
```sql
CREATE TABLE permissions (
    user_id INT PRIMARY KEY,
    username VARCHAR(50),
    permission_flags INT
);
```

```sql
-- Checking Read Permission
SELECT username,
```

```
    permission_flags & 4 AS has_read_permission,
    CASE WHEN permission_flags & 4 > 0 THEN 'Yes' ELSE 'No' END AS can_read
FROM permissions;
```

🔹 Toggling Execute Permission:
```sql
UPDATE permissions
SET permission_flags = permission_flags ^ 1
WHERE (permission_flags & 1) = 0;
```

📌 Bit Shifting Operations
Bit shifting shifts bits left (`<<`) or right (`>>`). Used for permission management, flag handling, and bitmask operations.

1. **Left Shift (`<<`) → Multiply by 2**
   ```sql
   0010 (2) << 1 → 0100 (4)
   ```

2. **Right Shift (`>>`) → Divide by 2**
   ```sql
   0100 (4) >> 1 → 0010 (2)
   ```

SQL Example:
```sql
SELECT id, value, value << 1 AS left_shift_1, value << 2 AS left_shift_2 FROM bit_shift_demo;
```

📌 EXISTS Keyword
EXISTS checks if a subquery returns rows.

```sql
SELECT name FROM customers c
WHERE EXISTS (
    SELECT * FROM orders o WHERE o.customerID = c.customerID
);
```

📌 CASE in SQL
Categorizing customers based on credit limit:
```sql
SELECT Name,
      CASE
         WHEN credit_limit >= 5000 THEN 'Premium'
         WHEN credit_limit >= 3000 THEN 'Gold'
         ELSE 'Standard'
      END AS CustomerTier
FROM Customers;
```

📌 RANK() / DENSE RANK()
Assigns rank to rows with possible ties.
```sql
SELECT name, department, salary,
      RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS SalaryRank,
      DENSE_RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS
SalaryDenseRank
FROM employees;
```

📌 LAG Function
Retrieves previous row values for trend analysis.
```sql
SELECT date_format(orderdate, '%Y-%m') AS yearmonth,
      SUM(totalamount) AS total_sales,
      LAG(SUM(totalamount)) OVER (ORDER BY date_format(orderdate, '%Y-%m')) AS
prev_month_sales,
      (SUM(totalAmount) - LAG(SUM(totalAmount)) OVER (ORDER BY date_format(orderdate,
'%Y-%m'))) * 100 /
      LAG(SUM(totalAmount)) OVER (ORDER BY date_format(orderdate, '%Y-%m')) AS
MoM_Growth
FROM orders
GROUP BY yearmonth;
```

📌 LeetCode Practice Problems

🔹 1️⃣ Average Time of Process per Machine
```sql
SELECT machine_id,
    ROUND(SUM(CASE WHEN activity_type = 'end' THEN timestamp ELSE -timestamp END)
/ COUNT(DISTINCT process_id), 3) AS processing_time
FROM Activity
GROUP BY machine_id;
```
🔗 [Problem Link](https://leetcode.com/problems/average-time-of-process-per-machine/)

🔹 2️⃣ Employee Bonus
```sql
SELECT e.name, b.bonus FROM Employee e LEFT JOIN Bonus b ON e.empId = b.empId
WHERE b.bonus < 1000 OR b.bonus IS NULL;
```
🔗 [Problem Link](https://leetcode.com/problems/employee-bonus/)

🔹 3️⃣ Students and Examinations
```sql
WITH StudentSubjects AS (
  SELECT s.student_id, s.student_name, sub.subject_name
  FROM Students s CROSS JOIN Subjects sub
)
SELECT ss.student_id, ss.student_name, ss.subject_name, COUNT(e.subject_name) AS
attended_exams
FROM StudentSubjects ss
LEFT JOIN Examinations e ON ss.student_id = e.student_id AND ss.subject_name =
e.subject_name
GROUP BY ss.student_id, ss.student_name, ss.subject_name;
```
🔗 [Problem Link](https://leetcode.com/problems/students-and-examinations/)

🔹 4️⃣ Managers with At Least 5 Direct Reports
```sql
SELECT e.name FROM Employee m JOIN Employee e ON m.managerId = e.id GROUP BY
e.id, e.name HAVING COUNT(m.id) >= 5;
```
🔗 [Problem Link](https://leetcode.com/problems/managers-with-at-least-5-direct-reports/)

🔹 5️⃣ Confirmation Rate

```sql
SELECT s.user_id, ROUND(COALESCE(SUM(CASE WHEN c.action = 'confirmed' THEN 1
ELSE 0 END) / COUNT(c.user_id), 0), 2) AS confirmation_rate
FROM Signups s LEFT JOIN Confirmations c ON s.user_id = c.user_id GROUP BY s.user_id;
```

🔗 [Problem Link](https://leetcode.com/problems/confirmation-rate/)

🚀 Conclusion
This README provides an in-depth look into **binary operations, bit shifting, EXISTS, CASE, ranking, LAG functions,** and SQL interview questions. Keep practicing! 🎯