Advanced SQL

Agenda

- Group By with Rollup
- Common Table Expressions
- Derived Tables
- Window Functions

Group By with Rollup

- Display deptwise total sal along with total sal of all employees.

```
+--------+----------+
| deptno | SUM(sal) |
+--------+----------+
|     20 | 10875.00 |
|     30 |  9400.00 |
|     10 |  8750.00 |
+--------+----------+
|        | 29025.00 |
+--------------------+
```

```
SELECT deptno, SUM(sal) FROM emp
GROUP BY deptno;

SELECT SUM(sal) FROM emp;

(SELECT deptno, SUM(sal) FROM emp
GROUP BY deptno)
```

```
UNION
(SELECT NULL, SUM(sal) FROM emp);
```

```
SELECT deptno, SUM(sal) FROM emp
GROUP BY deptno
WITH ROLLUP;
```

- Find deptwise, jobwise total sal along with total & sub-totals.

```
SELECT deptno, job, SUM(sal),COUNT(empno)
FROM emp
GROUP BY deptno, job
WITH ROLLUP;

SELECT deptno, job, SUM(sal),COUNT(empno)
FROM emp
GROUP BY job, deptno
WITH ROLLUP;
```

```
(SELECT deptno, job, SUM(sal),COUNT(empno)
FROM emp
GROUP BY deptno, job
WITH ROLLUP)
UNION
(SELECT deptno, job, SUM(sal),COUNT(empno)
FROM emp
GROUP BY job, deptno
WITH ROLLUP);
```

```
-- Works in Oracle.
SELECT deptno, job, SUM(sal),COUNT(empno)
FROM emp
GROUP BY CUBE(deptno,job);
```

```
SELECT deptno, job, SUM(sal) sumsal,COUNT(empno) cnt, GROUPING(deptno)
FROM emp
GROUP BY job, deptno
WITH ROLLUP;

SELECT deptno, job, SUM(sal) sumsal,COUNT(empno) cnt
FROM emp
GROUP BY job, deptno
WITH ROLLUP
HAVING GROUPING(deptno)=1;
```

Sub-query

- Find emps with max sal from each dept.

```
SELECT deptno, MAX(sal) FROM emp
GROUP BY deptno;

SELECT deptno, ename, sal FROM emp
WHERE sal IN (SELECT MAX(sal) FROM emp
GROUP BY deptno);
-- results not accurate

-- Assignment: try solving using coorelated sub-query.
```

Derived Tables

- Find emps with max sal from each dept.

```
SELECT deptno, MAX(sal) FROM emp
GROUP BY deptno;

SELECT ename, deptno, sal FROM emp;

SELECT e.ename, e.deptno, e.sal FROM emp e
INNER JOIN
(SELECT deptno, MAX(sal) maxsal FROM emp
GROUP BY deptno) md ON e.deptno = md.deptno
WHERE e.sal = md.maxsal;
```

CTE (Non-Recursive)

- Find emps with max sal from each dept.

```
WITH md AS
(SELECT deptno, MAX(sal) maxsal FROM emp
GROUP BY deptno)
SELECT e.ename, e.deptno, e.sal FROM emp e
INNER JOIN md  ON e.deptno = md.deptno
WHERE e.sal = md.maxsal;
```

- Find number of emps in each category -- POOR (sal less than 1500), RICH (sal more than 2500), MIDDLE (sal between 1500 and 2500).

```
SELECT ename, sal, CASE
WHEN sal < 1500 THEN 'POOR'
WHEN sal > 2500 THEN 'RICH'
```

```
ELSE 'MIDDLE'
END categry
FROM emp;

WITH empcategry AS (
SELECT ename, sal, CASE
WHEN sal < 1500 THEN 'POOR'
WHEN sal > 2500 THEN 'RICH'
ELSE 'MIDDLE'
END categry
FROM emp)
SELECT categry, COUNT(ename) FROM empcategry
GROUP BY categry;
```

- Get average of deptwise total sal.

```
SELECT deptno, SUM(sal) sumsal FROM emp
GROUP BY deptno;

WITH dept_total AS (
SELECT deptno, SUM(sal) sumsal FROM emp
GROUP BY deptno
)
SELECT AVG(sumsal) FROM dept_total;
```

Window Functions

- Display empno, ename, sal of each emp along with total sal of all emps.

```
SELECT empno, ename, sal, SUM(sal) FROM emp;
-- error: cannot use group fns with individual columns
```

```
SELECT empno, ename, sal, (SELECT SUM(sal) FROM emp) total FROM emp;

SELECT deptno, ename, sal,
SUM(sal) OVER () total
FROM emp;
```

- Display empno, ename, sal of each emp along with total sal of all emps in his dept.

```
SELECT deptno, ename, sal,
(SELECT SUM(sal) FROM emp WHERE deptno=e.deptno) total
FROM emp e
ORDER BY deptno;

-- homework: solve using join

SELECT deptno, ename, sal,
SUM(sal) OVER (PARTITION BY deptno) total
FROM emp;
```

ROW_NUMBER() vs RANK() vs DENSE_RANK()

```
SELECT ROW_NUMBER() OVER () sr,
ename, sal, deptno
FROM emp;

SELECT
ROW_NUMBER() OVER (PARTITION BY deptno) sr,
ename, sal, deptno
FROM emp;
```

```
SELECT RANK() OVER () rnk,
ename, sal, deptno
FROM emp;

SELECT RANK() OVER (ORDER BY sal) rnk,
ename, sal, deptno
FROM emp;

SELECT
RANK() OVER (PARTITION BY deptno) rnk,
ename, sal, deptno
FROM emp;

SELECT
RANK() OVER (PARTITION BY deptno ORDER BY sal) rnk,
ename, sal, deptno
FROM emp;
```

```
SELECT
ROW_NUMBER() OVER (ORDER BY sal) sr,
RANK() OVER (ORDER BY sal) rnk,
ename, sal, deptno
FROM emp;

SELECT
ROW_NUMBER() OVER (ORDER BY sal) sr,
RANK() OVER (ORDER BY sal) rnk,
DENSE_RANK() OVER (ORDER BY sal) drnk,
ename, sal, deptno
FROM emp;
```

```
SELECT
ROW_NUMBER() OVER (PARTITION BY deptno ORDER BY sal) sr,
RANK() OVER (PARTITION BY deptno ORDER BY sal) rnk,
DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal) drnk,
ename, sal, deptno
FROM emp;
```

```
SELECT
ROW_NUMBER() OVER (wnd) sr,
RANK() OVER (wnd) rnk,
DENSE_RANK() OVER (wnd) drnk,
ename, sal, deptno
FROM emp
WINDOW wnd AS (PARTITION BY deptno ORDER BY sal);
```

- Find emp with 3rd lowest sal.

```
SELECT
ROW_NUMBER() OVER (ORDER BY sal) sr,
RANK() OVER (ORDER BY sal) rnk,
DENSE_RANK() OVER (ORDER BY sal) drnk,
ename, sal, deptno
FROM emp
WHERE (DENSE_RANK() OVER (ORDER BY sal)) = 3;
-- error: cannot use the window function 'dense_rank' in this context.'


WITH sorted_emp AS(
SELECT
ROW_NUMBER() OVER (ORDER BY sal) sr,
RANK() OVER (ORDER BY sal) rnk,
```

```
DENSE_RANK() OVER (ORDER BY sal) drnk,
ename, sal, deptno
FROM emp
)
SELECT * FROM sorted_emp
WHERE drnk = 3;

-- homework: find emp with 4th highest sal.
```

- Find emp with max sal per dept.

```
SELECT
ROW_NUMBER() OVER (wnd) sr,
RANK() OVER (wnd) rnk,
DENSE_RANK() OVER (wnd) drnk,
ename, sal, deptno
FROM emp
WINDOW wnd AS (PARTITION BY deptno ORDER BY sal DESC);

WITH dept_sorted_emps AS(
SELECT
ROW_NUMBER() OVER (wnd) sr,
RANK() OVER (wnd) rnk,
DENSE_RANK() OVER (wnd) drnk,
ename, sal, deptno
FROM emp
WINDOW wnd AS (PARTITION BY deptno ORDER BY sal DESC)
)
SELECT * FROM dept_sorted_emps
WHERE rnk = 1;

-- homework: find emps with 2nd highest sal in each dept.
```

LEAD() and LAG()

- Find difference between consecutive entries.

```
SELECT
ROW_NUMBER() OVER (wnd) sr,
RANK() OVER (wnd) rnk,
ename, deptno, sal,
LAG(sal) OVER(wnd) prevsal
FROM emp
WINDOW wnd AS (ORDER BY sal);

SELECT
ROW_NUMBER() OVER (wnd) sr,
RANK() OVER (wnd) rnk,
ename, deptno, sal,
LAG(sal) OVER(wnd) prevsal,
sal - LAG(sal) OVER(wnd) diff
FROM emp
WINDOW wnd AS (ORDER BY sal);
```

Moving Window

```
DROP TABLE IF EXISTS transactions;
CREATE TABLE transactions (accid INT, txdate DATETIME, amount DOUBLE);

INSERT INTO transactions VALUES
(1, '2000-01-01', 1000),
(1, '2000-01-02', 2000),
(1, '2000-01-03', -500),
(1, '2000-01-04', -300),
(1, '2000-01-05', 4000),
(1, '2000-01-06', -2000),
```

```
(1, '2000-01-07', -200),
(2, '2000-01-02', 3000),
(2, '2000-01-04', 2000),
(2, '2000-01-07', -1000),
(3, '2000-01-03', 2000),
(3, '2000-01-04', -1000),
(3, '2000-01-06', 500);

SELECT * FROM transactions;

SELECT
ROW_NUMBER() OVER (wnd) sr,
accid, txdate, amount,
SUM(amount) OVER (wnd) balance
FROM transactions
WINDOW wnd AS (PARTITION BY accid ORDER BY txdate);

-- display statement of accid=1
SELECT
ROW_NUMBER() OVER (wnd) sr,
accid, txdate, amount,
SUM(amount) OVER (wnd) balance
FROM transactions
WHERE accid=1
WINDOW wnd AS (PARTITION BY accid ORDER BY txdate);
```

CTE (Recursive)

```
void seq(int s, int e) {
    if(s <= e) {
        System.out.println(s);
        seq(s+1, e);
    }
}
```

```
WITH RECURSIVE seq(n) AS (
    (SELECT 1)              -- anchor (s)
    UNION ALL
    (SELECT n+1 FROM seq    -- recursive member
    WHERE n < 4)            -- base condn (e)
)
SELECT * FROM seq;
```

- Print all years in which emps were hired.

```
SELECT DISTINCT(YEAR(hire)) FROM emp;
```

- Print years in which emps hired from 1975 to 1985 using CTE.

```
WITH RECURSIVE years(yr) AS(
    SELECT 1975
    UNION
    SELECT yr+1 FROM years
    WHERE yr < 1985
)
SELECT yr FROM years;

WITH RECURSIVE years(yr) AS(
    SELECT 1975
    UNION
    SELECT yr+1 FROM years
    WHERE yr < 1985
)
```

```
SELECT yr FROM years
WHERE yr IN (SELECT YEAR(hire) FROM emp);
```

- Print years in which emps NOT hired from 1975 to 1985 using CTE.

```
WITH RECURSIVE years(yr) AS(
    SELECT 1975
    UNION
    SELECT yr+1 FROM years
    WHERE yr < 1985
)
SELECT yr FROM years
WHERE yr NOT IN (SELECT YEAR(hire) FROM emp);
```

- Print level of each emp. Consider president level as 1 and level of his reporting is level+1.

```
SELECT empno, ename, mgr, deptno FROM emp
ORDER BY mgr;
WITH RECURSIVE emp_hierarchy AS(
SELECT empno, ename, mgr, deptno, 1 AS lvl
FROM emp WHERE mgr IS NULL
UNION ALL
SELECT e.empno, e.ename, e.mgr, e.deptno, lvl+1 AS lvl
FROM emp e JOIN emp_hierarchy eh ON e.mgr = eh.empno
)
SELECT * FROM emp_hierarchy;
```

Further readings

- https://www.mysqltutorial.org/mysql-cte/
- https://www.mysqltutorial.org mysql-recursive-cte/

- https://www.mysqltutorial.org/mysql-window-functions/
    - You may also refer syntax in sequence for other window functions.
- https://www.red-gate.com/simple-talk/sql/learn-sql-server/window-functions-in-sql-server-part-2-the-frame/