Advanced SQL

Agenda

- Sub-Queries
- GROUP BY WITH ROLLUP
- Derived Tables
- Common Table Expressions - Non-recursive
- Window Functions
- Common Table Expressions - Recursive

Getting Started

- terminal> mysql -u nilesh -pnilesh

```
SHOW DATABASES;

USE advsql;

SHOW TABLES;
-- emp, dept
```

Sub-Queries

- Dislay empno, name, sal of each emp along with total sal of all emps and percentage of his sal. Expected output: empno, ename, sal, totalsal, percent

```
SELECT empno, ename, sal FROM emp;

SELECT SUM(sal) FROM emp;

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

SELECT empno, ename, sal,
(SELECT SUM(sal) FROM emp) total,
100 * sal / (SELECT SUM(sal) FROM emp) percent
FROM emp;
```

- Dislay empno, name, sal of each emp along with total sal of all emps in his dept and percentage of his sal. Expected output: empno, ename, sal, depttotal, percent

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

```
SELECT empno, ename, sal FROM emp;

SELECT e.empno, e.ename, e.sal,
(SELECT SUM(sal) FROM emp WHERE deptno = e.deptno) depttotal
FROM emp e;

SELECT e.empno, e.ename, e.sal,
(SELECT SUM(sal) FROM emp WHERE deptno = e.deptno) depttotal,
100 * e.sal / (SELECT SUM(sal) FROM emp WHERE deptno = e.deptno) percent
FROM emp e;
```

GROUP BY WITH ROLLUP

- Display dept-wise total sal along with total sal of all emps.

```
+--------+----------+
| 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);
```

- Display dept-wise total sal along with total sal of all emps (with ROLLUP).

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

- Display dept-wise, job-wise total sal and emp count along with total sal of all emps, emp count. Also display (dept-wise) grand-total & sub-total.

```sql
SELECT deptno, job, SUM(sal) total, COUNT(empno) cnt
FROM emp
GROUP BY deptno, job;

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

- Display dept-wise, job-wise total sal and emp count along with total sal of all emps, emp count. Also display (job-wise) grand-total & sub-total.

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

- Display dept-wise, job-wise total sal and emp count along with total sal of all emps, emp count. Also display (dept-wise and job-wise) grand-total & sub-total.

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

SELECT deptno, job, SUM(sal) total, COUNT(empno) cnt
FROM emp
GROUP BY CUBE(deptno, job);
-- show all possible sub-total and grand-total
-- not supported in mysql. supported in oracle.
```

- GROUPING() Function

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

SELECT deptno, job, SUM(sal) total, COUNT(empno) cnt,
GROUPING(deptno) grdept, GROUPING(job) grjob
FROM emp
GROUP BY deptno, job
WITH ROLLUP;


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

Derived Tables

- Categorize emps in 3 categories
  - Poor < 1500
  - Rich > 2500
  - 1500 <= Middle <= 2500

```
SELECT empno, ename, sal, CASE
WHEN sal < 1500 THEN 'Poor'
WHEN sal > 2500 THEN 'Rich'
ELSE 'Middle'
END category
FROM emp;
```

- Count emps in each category.

```
CREATE VIEW vempcategory AS
SELECT empno, ename, sal, CASE
WHEN sal < 1500 THEN 'Poor'
WHEN sal > 2500 THEN 'Rich'
ELSE 'Middle'
END category
FROM emp;

SELECT * FROM vempcategory;

SELECT category, COUNT(1) cnt
FROM vempcategory
GROUP BY category;
```

- Count emps in each category (without views).

```
SELECT category, COUNT(1) cnt
FROM (
SELECT empno, ename, sal, CASE
WHEN sal < 1500 THEN 'Poor'
WHEN sal > 2500 THEN 'Rich'
ELSE 'Middle'
END category
FROM emp
) empcategory
GROUP BY category;
```

- Find max sal per dept.

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

- Find emps with max sal per dept.

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

SELECT empno, ename, deptno, sal FROM emp;

SELECT empno, ename, deptno, sal FROM emp
WHERE sal IN (
SELECT MAX(sal) FROM emp
GROUP BY deptno
);
-- not perfect -- homework (using subquery)

SELECT e.empno, 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;
```

Common Table Expressions - Non-recursive

- Find emps with max sal per dept (with CTE).

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

- Count emps in each category (without views).

```
WITH empcategory AS (
SELECT empno, ename, sal, CASE
WHEN sal < 1500 THEN 'Poor'
WHEN sal > 2500 THEN 'Rich'
ELSE 'Middle'
END category
FROM emp
)
SELECT category, COUNT(1) cnt
FROM empcategory
GROUP BY category;
```

- Get average of dept-wise total sal.

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

SELECT deptno, AVG(SUM(sal)) total FROM emp
GROUP BY deptno;
-- error: can't use group fn in group fn (in MySQL)

WITH depttotal AS (
SELECT deptno, SUM(sal) total FROM emp
GROUP BY deptno
)
SELECT AVG(total) FROM depttotal;

-- homework -- solve using Derived table
```

Window Functions

- Dislay empno, name, sal of each emp along with total sal of all emps.

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

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

```
FROM emp;

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

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

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

-- homework -- get same output with subquery
-- homework -- get same output with group by & join
```

- ROW_NUMBER() vs RANK() vs DENSE_RANK()
    - Meaningful only when window is sorted.

```
SELECT empno, ename, deptno, sal sFROM emp;

SELECT empno, ename, deptno, sal,
ROW_NUMBER() OVER () sr
FROM emp;
-- serial number -- random basis

SELECT empno, ename, deptno, sal,
ROW_NUMBER() OVER (ORDER BY sal) sr
FROM emp;
-- serial number -- sorted order of sal asc

SELECT empno, ename, deptno, sal,
ROW_NUMBER() OVER (ORDER BY sal DESC) sr
FROM emp;
-- serial number -- sorted order of sal desc

SELECT empno, ename, deptno, sal,
ROW_NUMBER() OVER (PARTITION BY deptno) sr
FROM emp;
-- serial number -- random basis - dept wise

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

SELECT empno, ename, deptno, sal,
RANK() OVER () rnk
FROM emp;
-- rank() Meaningful only when used with order by
-- otherwise all records treated with same rank (1)

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


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

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

- Find emp with 5th lowest sal (using window fn).

```

SELECT empno, ename, deptno, sal,
ROW_NUMBER() OVER (ORDER BY sal) sr,
RANK() OVER (ORDER BY sal) rnk,
DENSE_RANK() OVER (ORDER BY sal) drnk
FROM emp
WHERE (DENSE_RANK() OVER (ORDER BY sal)) = 5;
-- ERROR 3593 (HY000): You cannot use the window function 'dense_rank' in this
context.'

WITH sortemp AS(
SELECT empno, ename, deptno, sal,
ROW_NUMBER() OVER (ORDER BY sal) sr,
RANK() OVER (ORDER BY sal) rnk,
DENSE_RANK() OVER (ORDER BY sal) drnk
FROM emp
)
SELECT * FROM sortemp
WHERE drnk = 5;
```

- ROW_NUMBER() vs RANK() vs DENSE_RANK()

```

SELECT empno, ename, deptno, sal,
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
FROM emp;

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

- Find emps with highest sal in each dept.

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

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

- homework -- Find emps with second highest sal in each dept.
- Difference between consecutive entries

```
SELECT empno, ename, deptno, sal
FROM emp
ORDER BY sal;

SELECT empno, ename, deptno,
sal,
LEAD(sal) OVER (wnd) nextsal,
ROW_NUMBER() OVER (wnd) sr
FROM emp
WINDOW wnd AS (ORDER BY sal);

SELECT empno, ename, deptno,
sal,
LEAD(sal) OVER (wnd) nextsal,
LEAD(sal) OVER (wnd) - sal diff,
ROW_NUMBER() OVER (wnd) sr
FROM emp
WINDOW wnd AS (ORDER BY sal);
```

- Display mini-statement of bank account along with running balance.

```sql
DROP TABLE IF EXISTS transactions;

CREATE TABLE transactions (accid INT, txdate DATETIME, amount DOUBLE);
INSERT INTO transactions VALUES (1, '2000-01-01', 1000);
INSERT INTO transactions VALUES (1, '2000-01-02', 2000);
INSERT INTO transactions VALUES (1, '2000-01-03', -500);
INSERT INTO transactions VALUES (1, '2000-01-04', -300);
INSERT INTO transactions VALUES (1, '2000-01-05', 4000);
INSERT INTO transactions VALUES (1, '2000-01-06', -2000);
INSERT INTO transactions VALUES (1, '2000-01-07', -200);
INSERT INTO transactions VALUES (2, '2000-01-02', 3000);
INSERT INTO transactions VALUES (2, '2000-01-04', 2000);
INSERT INTO transactions VALUES (2, '2000-01-07', -1000);
INSERT INTO transactions VALUES (3, '2000-01-03', 2000);
INSERT INTO transactions VALUES (3, '2000-01-04', -1000);
INSERT INTO transactions VALUES (3, '2000-01-06', 500);

SELECT * FROM transactions
ORDER BY accid, txdate;

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);

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);
```

RECURSIVE CTE

- print numbers 1 to 4 as a table.

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

```
WITH RECURSIVE cte_seq(n) AS
(
    SELECT 1                    -- anchor member
    UNION ALL
    SELECT n+1 FROM cte_seq     -- recursive member
    WHERE n <= 3                -- base condition
)
SELECT * FROM cte_seq;
```

- Print years from 1980 to 1990 where no emp is hired.

```
SELECT YEAR(hire) yr, COUNT(empno) cnt FROM emp
GROUP BY yr;

WITH RECURSIVE years(yr) AS
(
    SELECT 1980
    UNION ALL
    SELECT yr+1 FROM years WHERE yr < 1990
)
SELECT * FROM years;

WITH RECURSIVE years(yr) AS
(
    SELECT 1980
    UNION ALL
    SELECT yr+1 FROM years WHERE yr < 1990
)
SELECT yr FROM years
WHERE yr NOT IN (SELECT YEAR(hire) FROM emp);
```

- Print level of each emp in the company. Consider level of PRESIDENT as 1 and level number increases hierarchially.

```
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 Reading

- 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/