Agenda

- GROUP BY clause
- HAVING clause
- Joins

MultiRow functions

- Also called as Group Functions,Aggegrate Functions
- These functions are executed for a group of rows
- It produces single result for group of input
- n input rows -> 1 output row

```
-- display count of employees
SELECT COUNT(*) FROM emp;
SELECT COUNT(empno) FROM emp;

-- display total salary expenditure on all emps
SELECT SUM(sal) FROM emp;

--display highest salary
SELECT MAX(sal) FROM emp;

--display lowest salary
SELECT MIN(sal) FROM emp;

-- display avg salary paid to emps
SELECT AVG(sal) FROM emp;

-- display count of emps getting commission
SELECT COUNT(comm) FROM emp;
-- null values are ignored in group functions
```

Limitations of Group Functions

```
-- to check for limitations change sql mode
SET @@sql_mode='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES NO_ENGINE_SUBSTITUTION';

SELECT ename, MAX(sal) FROM emp;
-- error - cannot select column with group function

SELECT LOWER(ename),MAX(sal) FROM emp;
--error - cannot use single row functions  with group functions

SELECT * FROM emp WHERE sal = MAX(sal);
--error - cannot use group functions in where clause
```

```
SELECT SUM(MAX(sal)) FROM emp;
-- error - cannot nest group functions
```

Group By Clause

- Group functions work on group of rows
- with group by clause we can use group functions on specified group of rows

```
-- display deptwise total salary spent
SELECT deptno,SUM(sal) FROM emp GROUP BY deptno;

-- display jobwise count of employees
SELECT job,COUNT(empno) FROM emp GROUP BY job;

-- display deptwise Max salary
SELECT deptno,MAX(sal) FROM emp GROUP BY deptno;

-- display count of emp in every dept jobwise
SELECT deptno,job FROM emp ORDER BY deptno,job;
SELECT deptno,job,COUNT(empno) FROM emp GROUP BY deptno,job;
SELECT deptno,job,COUNT(empno) FROM emp GROUP BY deptno,job ORDER BY deptno,job;
```

Having Clause

- It must be used with Group By Clause only
- Used to apply condition on aggegrate values
- Having vs WHERE
    - Where clause evaluates for every row
    - Having clause evaluates for group
    - Where can be used with columns, single row function but not with group functions
    - Having is used with group columns but not with other columns

```
-- display deptno in which total sal > 9000
SELECT deptno,SUM(sal) FROM emp GROUP BY deptno;
SELECT deptno,SUM(sal) AS total FROM emp GROUP BY deptno HAVING total>9000;

-- display jobs for which avg sal is more than 2500.
SELECT job,AVG(sal) FROM emp GROUP BY job HAVING AVG(sal)>2500;

-- display max sal of emp from dept 20 and 30
SELECT deptno,MAX(sal) FROM emp GROUP BY deptno;
SELECT deptno,MAX(sal) FROM emp WHERE deptno IN (10,20) GROUP BY deptno;
-- more efficient

SELECT deptno,MAX(sal) FROM emp GROUP BY deptno HAVING deptno IN(10,20);
-- less efficient
```

```
-- display the deptno that spends the lowest on emp salary
SELECT deptno,SUM(sal) FROM emp GROUP BY deptno;
SELECT deptno,SUM(sal) FROM emp GROUP BY deptno ORDER BY SUM(sal);
SELECT deptno,SUM(sal) FROM emp GROUP BY deptno ORDER BY SUM(sal) LIMIT 1;

-- find the jobs which have highest avg sal.
SELECT job, AVG(sal) FROM emp GROUP BY job ORDER BY AVG(sal) DESC LIMIT 1;
```

Table Relations

- To avoid redundancy of the data, data should be organized into multiple tables so that tables are related to each other.
- The relations can be one of the following
    - 1. One to One
    - 2. One to Many
    - 3. Many to One
    - 4. Many to Many
- Entity relations is outcome of Normalization process.
- We need to apply Multi-Table Joins

Joins

- It is used to fetch the data from multiple tables in single query
- Types of joins are
    - 1. CROSS JOIN
    - 2. INNER JOIN
    - 3. LEFT OUTER JOIN
    - 4. RIGHT OUTER JOIN
    - 5. FULL OUTER JOIN
    - 6. SELF JOIN

```
-- use joins.sql for all the join demos
SOURCE <path to joins.sql file>

SELECT * FROM emps;
SELECT * FROM depts;
SELECT * FROM addr;
SELECT * FROM meeting;
SELECT * FROM emp_meeting;
```

1. CROSS JOIN

```
SELECT ename,dname FROM emps CROSS JOIN depts;

SELECT e.ename,d.dname FROM emps e CROSS JOIN depts d;
```

```
SELECT e.ename,d.dname FROM depts d CROSS JOIN emps e;

SELECT e.ename,d.dname FROM emps AS e CROSS JOIN depts AS d;
-- using AS in table alias is optional

SELECT emps.ename,depts.dname FROM emps CROSS JOIN depts;
-- can use table name directly if table alias are not given
```

2. INNER JOIN

- The inner JOIN is used to return rows from both tables that satisfy the join condition.
- Non-matching rows from both tables are skipped.
- If join condition contains equality check, it is referred as equi-join,otherwise it is non-equi-join.

```
-- display empname and his deptname
SELECT e.ename,d.dname FROM emps e
INNER JOIN depts d ON e.deptno=d.deptno;
-- called as equi-joins

-- display empname and the deptnames in which he is not working
SELECT e.ename,d.dname FROM emps e
INNER JOIN depts d ON e.deptno!=d.deptno;
-- called as non-equi joins
```

LEFT OUTER JOIN

- Common data from both table, plus the data from left side table
- Left outer join is used to return matching rows from both tables along with additional rows in left table.
- Corresponding to additional rows in left table, right table values are taken as NULL.

```
-- display empname and his deptname.
SELECT e.ename,d.dname FROM emps e
LEFT OUTER JOIN depts d ON e.deptno=d.deptno;

SELECT e.ename,d.dname FROM emps e
LEFT JOIN depts d ON e.deptno=d.deptno;
-- using outer keyword is optional
```

RIGHT OUTER JOIN

- Common data from both table, plus the data from right side table
- Right outer join is used to return matching rows from both tables along with additional rows in right table.
- Corresponding to additional rows in right table, left table values are taken as NULL.

```
-- display empname and his deptname.
SELECT e.ename,d.dname FROM emps e
RIGHT OUTER JOIN depts d ON e.deptno=d.deptno;

SELECT e.ename,d.dname FROM emps e
RIGHT JOIN depts d ON e.deptno=d.deptno;
-- using outer keyword is optional
```

Full Outer Join

- Full join is used to return matching rows from both tables along with additional rows in both tables.
- Corresponding to additional rows in left or right table, opposite table values are taken as NULL.
- Full outer join is not supported in MySQL, but can be simulated using set operators.

```
SELECT e.ename, d.dname FROM emps e
FULL OUTER JOIN depts d ON e.deptno = d.deptno;
-- NOT SUPPORTED
```

Set Operators

- Used to combine results of two queries (if output contains same number of columns).

```
(SELECT dname AS name FROM depts)
UNION ALL
(SELECT ename FROM emps);

(SELECT sal FROM emp)
UNION ALL
(SELECT price FROM books);

(SELECT e.ename, d.dname FROM emps e
LEFT OUTER JOIN depts d ON e.deptno = d.deptno)
UNION ALL
(SELECT e.ename, d.dname FROM emps e
RIGHT OUTER JOIN depts d ON e.deptno = d.deptno);

(SELECT e.ename, d.dname FROM emps e
LEFT OUTER JOIN depts d ON e.deptno = d.deptno)
UNION
(SELECT e.ename, d.dname FROM emps e
RIGHT OUTER JOIN depts d ON e.deptno = d.deptno);
-- simulation of full outer join in MySQL
```

Self Join

```sql
-- display ename and manager name from emps;
SELECT e.ename, m.ename AS mname FROM emps e
INNER JOIN emps m ON e.mgr = m.empno;

SELECT e.ename, m.ename AS mname FROM emps e
LEFT JOIN emps m ON e.mgr = m.empno;
```