MySQL - RDBMS

Agenda

- SQL Functions
  - Numeric Functions
  - Date and Time Functions
  - Control Flow Functions
  - Misc/Info Functions
  - List Functions
  - NULL Functions
  - Group Functions
- GROUP BY clause
- HAVING clause

SQL Functions

Numeric Functions

```
SELECT USER(), DATABASE();
-- | sunbeam@localhost | classwork  |

HELP Numeric Functions;

SELECT POWER(2, 5);

SELECT SQRT(2);

SELECT RAND();

-- fetch rows in random order
SELECT empno, ename, sal FROM emp
ORDER BY RAND();

SELECT PI();

SELECT ROUND(3.141593, 2), ROUND(3.141593, 4);
-- 3.14, 3.1416

SELECT ROUND(314159.3, -2), ROUND(31415.93, -2);
-- 314200, 31400

SELECT ROUND(3.141593, -2);
-- 0

SELECT ROUND(7246851749, -5);

SELECT * FROM books;
```

```
SELECT id, name, ROUND(price,2) FROM books;
```

- CEIL -- nearest higher integer.
- FLOOR -- nearest lower integer.

```
SELECT CEIL(3.14), FLOOR(3.14);
-- 4, 3

SELECT CEIL(-3.14), FLOOR(-3.14);
-- -3, -4
```

Date and Time Functions

- DATE -- '1000-01-01' to '9999-12-31'
- TIME -- '-838:59:59' to '838:59:59'
- DATETIME -- '1000-01-01 00:00:00.000000' to '9999-12-31 23:59:59.999999'
- TIMESTAMP -- '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC
- YEAR -- '1901' to '2155'

```
HELP Date and Time Functions;

SELECT NOW(), SLEEP(5), SYSDATE();

SELECT DATE('2000-05-24 14:47:20'), TIME('2000-05-24 14:47:20');

SELECT DATE(NOW()), TIME(NOW());

SELECT DATE_ADD(NOW(), INTERVAL 4 DAY);

SELECT DATE_ADD(NOW(), INTERVAL 1 MONTH);

SELECT DATEDIFF(NOW(), '1983-09-28');
-- return number of days

SELECT TIMESTAMPDIFF(YEAR, '1983-09-28', NOW());
-- return number of years

SELECT ename, hire, TIMESTAMPDIFF(YEAR, hire, NOW()) exp_yrs, TIMESTAMPDIFF(MONTH,
hire, NOW()) exp_mons FROM emp;

SELECT ename, hire, TIMESTAMPDIFF(YEAR, hire, NOW()) exp_yrs, TIMESTAMPDIFF(MONTH,
hire, NOW()) % 12 exp_mons FROM emp;

SELECT DATE_FORMAT(NOW(), '%d-%b-%Y');
-- 07-Oct-2021

SELECT DAY(NOW()), MONTH(NOW()), YEAR(NOW()), HOUR(NOW()), MINUTE(NOW()),
```

```
SECOND(NOW()), WEEKDAY(NOW());

-- find all emps hired in 1982
SELECT * FROM emp WHERE YEAR(hire) = 1982;

-- MySQL standard date format: 'yyyy-mm-dd'
-- input date: 'dd/mm/yyyy' -- ??
SET @str = '28/09/2021';
SELECT STR_TO_DATE(@str, '%d/%m/%Y');
```

Control Functions

- IF(condition, expr_if_true, expr_if_false).

```
HELP IF FUNCTION;

-- display ename, sal and category
-- category = RICH if sal > 2500
-- category = POOR if sal <= 2500
SELECT ename, sal, IF(sal > 2500, 'RICH', 'POOR') AS category FROM emp;

-- print number is +ve or -ve or zero.
SET @num = 2;
-- MySQL user-defined variable -- session scope.
-- when MySQL CLI exit, variable will be destroyed

SELECT @num;

SELECT IF(@num > 0, '+ve', IF(@num < 0, '-ve', 'zero'));
```

List Functions

```
SELECT CONCAT('A', 12, 'B', 34.45);

SELECT GREATEST(23, 98, 53, 67);

SELECT LEAST(23, 98, 53, 67);

SELECT name, price, LEAST(price,700) FROM books;

SELECT GREATEST('AEROPLANE', 'CAR');

SELECT LEAST('AEROPLANE', NULL, 'CAR');

SELECT CONCAT('A', 12, NULL, 'B', 34.45);
```

Misc Functions

```
SELECT VERSION();

SELECT SYSDATE();

SELECT USER(), DATABASE();
```

NULL Functions

- NULL is special value in RDBMS.
- It is irrespective of data type.
- NULL is not 0, 0.0, '\0', 'NULL'.
- NULL represent missing/absent/empty value.

```
SELECT COALESCE(NULL, NULL, 12, 'Nilesh');
-- return = first non-null value.

SELECT COALESCE(12.34, NULL, 'Nilesh');

-- display comm of emp and if no comm then display sal.
SELECT ename, comm, sal, COALESCE(comm, sal) FROM emp;

SELECT ename, comm, sal, IF(comm IS NULL, sal, comm) FROM emp;

SELECT ename, comm, sal, IFNULL(comm, sal) FROM emp;
-- if arg1 == NULL, then result = arg2, else arg1.

SELECT ename, sal, NULLIF(sal,3000.0) FROM emp;
-- if arg1 == arg2, then result = NULL, else arg1.

SELECT IFNULL(NULL, 'Hello');
```

Group Functions

- Single Row Functions:
  - "n" Input Rows --> "n" Output Rows
  - Function execute once for each row.
- Group Functions/Multi Row Functions/Aggregate Functions
  - "n" Input Rows --> "1" Output Row
  - Aggregate value
    - COUNT(), SUM(), AVG(), MAX(), MIN()
    - STDEV(), COR(), ...

```
SELECT COUNT(sal), SUM(sal), AVG(sal), MAX(sal), MIN(sal) FROM emp;

SELECT COUNT(comm), SUM(comm), AVG(comm), MAX(comm), MIN(comm) FROM emp;
-- NULL values are ignored by Group Functions.
```

```
-- display max income and min income from emp.
-- income = sal + comm
SELECT ename, sal, comm, sal + IFNULL(comm,0) AS income FROM emp;

SELECT MAX(sal + IFNULL(comm,0)), MIN(sal + IFNULL(comm,0)) FROM emp;
```

- GREATEST vs MAX and LEAST vs MIN
  - GREATEST/LEAST -- single row function.
  - MAX/MIN -- group/aggregate function.
  - GREATEST/LEAST -- operate on multiple values from the same row.
  - MAX/MIN -- operate on multiple values in different rows (given column).
  - GREATEST/LEAST -- list function (multiple args).
  - MAX/MIN -- single arg (column name)
  - GREATEST/LEAST -- if any arg is NULL, result is NULL.
  - MAX/MIN -- if any row has NULL value in given column, that will be ignored.

LIMITATIONS OF GROUP FUNCTIONS

```
SET @@sql_mode='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION';

SELECT ename, MAX(sal) FROM emp;
-- error: cannot select any column with group fn.

SELECT LOWER(ename), MAX(sal) FROM emp;
-- error: cannot select single row fn with group fn.

SELECT * FROM emp WHERE sal = MAX(sal);
-- error: cannot use group fn in WHERE clause

SELECT SUM(MAX(sal)) FROM emp;
-- error: cannot nest group fn in each other.
```

- To set sql_mode permanently.
  - step 1: Run Notepad -- "Run as Administrator".
  - step 2: Open my.ini from C:\ProgramData\MySQL\MySQL Server 8.0.
  - step 3: Under [mysqld] change sql_mode.
    - sql-mode=ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
  - step 4: Restart MySQL server (or restart computer).

DQL - SELECT

GROUP BY clause

- By default GROUP functions work on all the rows.
- With GROUP BY we can use GROUP functions on group of rows.

```
SELECT deptno, COUNT(sal), SUM(sal), AVG(sal), MAX(sal), MIN(sal) FROM emp
GROUP BY deptno;

SELECT job, COUNT(sal), SUM(sal), AVG(sal), MAX(sal), MIN(sal) FROM emp
GROUP BY job;

SELECT deptno, COUNT(empno) FROM emp
GROUP BY deptno;

SELECT job, COUNT(empno) FROM emp
GROUP BY job;
```

```
SELECT empno, ename, 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;

-- deptno   job     count
--  10      C       1
--  10      M       1
--  10      P       1
--  20      A       2
--  20      C       2
--  20      M       1
--  30      C       1
--  30      M       1
--  30      S       4
```

```
SELECT empno, COUNT(empno) FROM emp
GROUP BY empno;
```

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

SELECT SUM(sal) FROM emp GROUP BY deptno;
-- it is not mandetory to project grouped column.
-- however output will be meaningless.

SELECT deptno, SUM(sal) FROM emp;
-- error: cannot select column with group fn.
```

HAVING clause

- Must be used with GROUP BY clause only.
- Mainly used to apply condition on aggregate values/results.
- HAVING clause vs WHERE clause
    - WHERE clause: evaluated for each row.
    - HAVING clause: evaluated for each group.
    - WHERE clause: can be used with column, single row fn, but not group fn.
    - HAVING clause: can be used with grouped column or group fn, but not on other columns.

```
-- display deptno in which total sal is more than 9000.
SELECT deptno, SUM(sal) FROM emp
GROUP BY deptno
HAVING SUM(sal) > 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 for each job for emps in deptno 10 and 20.
SELECT * FROM emp WHERE deptno IN (10,20);

SELECT job, MAX(sal) FROM emp
WHERE deptno IN (10,20)
GROUP BY job;

-- display max sal for each job for emps in deptno 10 and 20. display max sal only
if it is more than 2500.

SELECT job, MAX(sal) FROM emp
WHERE deptno IN (10,20)
GROUP BY job
HAVING MAX(sal) > 2500;
```

```
-- find avg sal for deptno 10 and 20.
SELECT deptno, AVG(sal) FROM emp
WHERE deptno IN (10, 20)
GROUP BY deptno;
-- more efficient

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

```
-- find the dept that spends max on emp sals.
SELECT deptno, SUM(sal) FROM emp
GROUP BY deptno;

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

SELECT deptno, SUM(sal) FROM emp
GROUP BY deptno
ORDER BY SUM(sal) DESC
LIMIT 1;
```

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

-- find the jobs which have lowest avg income.
SELECT job, AVG(sal + IFNULL(comm,0.0)) FROM emp
GROUP BY job
ORDER BY AVG(sal + IFNULL(comm,0.0))
LIMIT 1;
```