MySQL - RDBMS

Agenda

- Transaction internals
- SQL Functions
    - NULL values
- Group By clause
- Having clause
- REGEXP operator
- ~~Table Relations~~

Q & A

```
SELECT ename, hire, DATE_FORMAT(hire, '%W') FROM emp;

SELECT ROUND(5678.12, -4);
-- round upto 4 places before decimal --> 10000
-- 5678.12 >= 5000 --> 10000

SELECT ROUND(23493.12, -4);
-- 23493.12 < 2500 --> 20000

SELECT ename, hire, LAST_DAY(hire) FROM emp;

SELECT ename, REPLACE(ename, 'IT', 'XX') FROM emp;

SELECT ename, REPLACE(REPLACE(ename, 'I', 'X'), 'T', 'Y') FROM emp;

SELECT ename FROM emp WHERE LENGTH(ename) = 4;

-- SUBSTRING(string, start_pos, length);
-- start_pos is 1 based (not 0 based).
--      +ve value means from the start, and -ve value means from the end.
-- length is num of chars.
--      only +ve value allowed.
--      <=0 gives empty string
```

Transaction internals

SQL Functions

NULL related

- Most of SQL functions result NULL when NULL is passed as one of its arg.

```
SELECT SUBSTRING('Sunbeam', NULL, 4);

SELECT POWER(2, NULL);

SELECT DATE_FORMAT('2021-05-21', NULL);

-- print total income of emp.
SELECT ename, sal, comm, sal + comm AS income from emp;
```

- Few functions are designed to be used with NULL.
  - IFNULL(expr, value) --> if expr is NULL, consider this value.
    - IFNULL(comm, 0.0) --> if comm is NULL, consider it as 0.
  - NULLIF(expr, value) --> if expr is equal to given value, then consider result as NULL.
  - ISNULL(expr) --> returns 1 if null, else 0.
  - COALESCE(expr1, expr2, expr3) --> returns first non-null value.

```
SELECT ename, sal, comm, IFNULL(comm, 0.0) FROM emp;

SELECT ename, sal, comm, sal + IFNULL(comm, 0.0) AS income FROM emp;

SELECT ename, sal, NULLIF(sal, 3000) FROM emp;

SELECT ename, comm, NULLIF(comm, 0.0) FROM emp;

-- find emps who do not get comm (or comm is 0).
SELECT ename, comm FROM emp WHERE NOT (comm IS NULL OR comm = 0.0);

SELECT ename, comm FROM emp WHERE NULLIF(comm, 0.0) IS NOT NULL;

-- ISNULL function is similar to IS NULL operator.
SELECT ename, comm, ISNULL(comm) FROM emp;

SELECT ename, comm FROM emp WHERE ISNULL(comm) = 1;

SELECT ename, comm FROM emp WHERE comm IS NULL;

-- COALESCE
SELECT COALESCE(NULL, NULL, 1, 'A', '2021-02-12');

SELECT COALESCE(NULL, NULL, NULL, 'A', '2021-02-12');

-- if comm is null, return sal; otherwise return comm
SELECT ename, sal, comm, COALESCE(comm, sal) FROM emp;
```

List Functions (Single Row Fn)

- CONCAT(expr1, expr2, ...)
- COALESCE(expr1, expr2, ...)

- GREATEST(expr1, expr2, ...) --> return max from the list
- LEAST(expr1, expr2, ...) --> return min from the list

```
SELECT sal, comm, GREATEST(sal, IFNULL(comm,0.0)) AS gr FROM emp;

SELECT sal, comm, LEAST(sal, IFNULL(comm,0.0)) AS gr FROM emp;
```

Control Function

- IF(condition, true-expr, false-expr) --> similar to ternary operator

```
SELECT ename, sal, IF(sal >= 2500, 'Rich', 'Poor') FROM emp;
```

Multi-Row/Group Functions

- Aggregate operations
  - SUM(), AVG(), COUNT(), MAX(), MIN(), STDEV(), ...
- Aggregate operation on multiple values from multiple rows.
- SUM(sal) --> returns sum of sal of all emps (14 rows --> 14 sals)

```
SELECT SUM(sal) FROM emp;

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

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

- Limitations of Group Functions

```
SELECT @@sql_mode;

-- temp change (will be discarded when mysql client is closed)
SET sql_mode = 'ONLY_FULL_GROUP_BY';

SELECT @@sql_mode;

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

-- cannot select single row function with group function
SELECT LOWER(ename), SUM(sal) FROM emp;

-- find emp with max sal.
-- cannot use group function in WHERE clause.
SELECT * FROM emp WHERE sal = MAX(sal);
```

```
-- cannot call group function in another group function.
SELECT MAX(SUM(sal)) FROM emp;
```

- To make SQL mode permanent.
  - In C:\ProgramData\MySQL\MySQL Server 8.0\my.ini
    - Under [mysqld]
      - sql-mode="STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION,ONLY_FULL_GROUP_BY"
  - Restart the computer
  - Login to MySQL CLI.
    - SELECT @@sql_mode;

GROUP BY

- Allows me to SELECT grouped-column with GROUP functions

```
SELECT * FROM emp;

-- on which dept company is spending how much on salaries?
SELECT deptno, SUM(sal) FROM emp; -- error
SELECT deptno, SUM(sal) FROM emp GROUP BY deptno;

-- in which dept how many emps are there?
SELECT deptno, COUNT(empno) FROM emp GROUP BY deptno;

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

-- for each job how many emps are available?
SELECT job, COUNT(empno) FROM emp GROUP BY job;

-- how emps are hired in each year.
SELECT * FROM emp WHERE YEAR(hire) = 1980;
SELECT COUNT(empno) FROM emp WHERE YEAR(hire) = 1980;
SELECT COUNT(empno) FROM emp WHERE YEAR(hire) = 1981;
SELECT COUNT(empno) FROM emp WHERE YEAR(hire) = 1982;
SELECT COUNT(empno) FROM emp WHERE YEAR(hire) = 1983;

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

-- get max sal per job.
SELECT job, MAX(sal) FROM emp
GROUP BY job;

-- count num of emps per dept, per job.
SELECT DISTINCT deptno FROM emp;

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

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

SELECT DISTINCT deptno, job FROM emp;

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;
```

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

SELECT SUM(sal) FROM emp GROUP BY deptno;

SELECT deptno, SUM(sal) FROM emp; -- error

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

```
-- find the dept which spend max on sal of emps.
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 dept which spend max on sal of emps other managers.
SELECT * FROM emp WHERE job <> 'MANAGER';

SELECT SUM(sal) FROM emp WHERE job <> 'MANAGER'; -- total sal of emps other than
managers

SELECT deptno, SUM(sal) FROM emp WHERE job <> 'MANAGER'
GROUP BY deptno;

SELECT deptno, SUM(sal) FROM emp WHERE job <> 'MANAGER'
GROUP BY deptno
ORDER BY SUM(sal) DESC;

SELECT deptno, SUM(sal) FROM emp
WHERE job <> 'MANAGER'
GROUP BY deptno
ORDER BY SUM(sal) DESC
```

```
LIMIT 1;

SELECT deptno, SUM(sal) FROM emp
GROUP BY deptno
WHERE job <> 'MANAGER'
ORDER BY SUM(sal) DESC
LIMIT 1; -- error: SELECT (1) WHERE (2) GROUP BY (3) ORDER BY (4) LIMIT
```

Having clause

- To filter output of GROUP BY.
- HAVING clause MUST be written After GROUP BY.
- HAVING clause
  - filtering based on some condition.
  - filter output of GROUP BY.
  - condition on aggregate function.
  - must be after the GROUP BY.
- WHERE clause
  - filtering based on some condition.
  - filter rows of table.
  - condition on individual columns.
  - if present, must be before the GROUP BY.

```
-- find all jobs who have more than 3 empoyees.
SELECT job, COUNT(empno) FROM emp
GROUP BY job;

SELECT job, COUNT(empno) FROM emp
GROUP BY job
HAVING COUNT(empno) > 3;

-- find all depts who have avg sal more than 2500.0
SELECT deptno, AVG(sal) FROM emp
GROUP BY deptno;

SELECT deptno, AVG(sal) FROM emp
GROUP BY deptno
HAVING AVG(sal) > 2500.0;

-- get total sal of managers, analysts and clerks in deptwise for dept 10 & 20.
SELECT deptno, SUM(sal) FROM emp
WHERE job IN ('MANAGER', 'ANALYST', 'CLERK')
GROUP BY deptno;

SELECT deptno, SUM(sal) FROM emp
WHERE job IN ('MANAGER', 'ANALYST', 'CLERK')
GROUP BY deptno
HAVING deptno IN (10, 20);

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

```
WHERE job IN ('MANAGER', 'ANALYST', 'CLERK') AND deptno IN (10, 20)
GROUP BY deptno;

-- find num of emps per job for MANAGER and ANALYST.
SELECT job, COUNT(empno) FROM emp
GROUP BY job
HAVING job IN ('MANAGER', 'ANALYST');

SELECT job, COUNT(empno) FROM emp
WHERE job IN ('MANAGER', 'ANALYST')
GROUP BY job;

-- find jobs where avg sal < 2000.
SELECT job, AVG(sal) FROM emp
WHERE AVG(sal) < 2000
GROUP BY job; -- error

SELECT job, AVG(sal) FROM emp
GROUP BY job
HAVING AVG(sal) < 2000;
```

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

SELECT ename, sal FROM emp;
```

REGEXP operator

- Used in WHERE clause to filter the records.
- Similar to LIKE to filter based on some pattern (of string type).
  - Wildcard chars: %, _

```
-- find all emps whose name start with A
SELECT * FROM emp WHERE ename LIKE 'A%';
```

- REGEXP stands for Regular Expressions. Used at many places.
  - Database -- for searching/validation.
    - RDBMS, NoSQL
  - Programming languages
    - Python, JS, Java, C#, Perl, ...
  - Web Programming -- for validation
    - HTML, JS, PHP, Java, Python, .NET, ...
- REGEXP operator allows giving patterns using rich Wildcard.
  - Wildcard chars: $, ^, ., +, *, [], [^], {}, ...

- ^ <-- at the start.
- $ <-- at the end.
- . <-- any one char.
- [a-z] <-- scanset/range e.g. a to z
- [a-z0-9] <-- scanset/range e.g. a to z or 0 to 9
- [aiu] <-- scanset -- any one letter out of a, i and u
- [^...] <-- inverse of scanset
  - [^a-z] <-- anything other than alphabet
  - [^0-9] <-- anything other than digit
- (...) <-- group multiple chars

```
CREATE TABLE food(word VARCHAR(30));

INSERT INTO food VALUES ('this'), ('biscuit'), ('isnt'), ('tasty, '), ('but'),
('that'), ('cake'), ('is'), ('really good');

SELECT * FROM food;

-- get all words containing "is"
SELECT * FROM food WHERE word REGEXP 'is';

-- get all words starting with "is"
SELECT * FROM food WHERE word REGEXP '^is';

-- get all words ending with "is"
SELECT * FROM food WHERE word REGEXP 'is$';

-- get all words having only is.
SELECT * FROM food WHERE word REGEXP '^is$';
```

```
CREATE TABLE selection(word VARCHAR(30));

INSERT INTO selection VALUES ('bag'), ('beg'), ('big'), ('bog'), ('bug'), ('b*g'),
('bg'), ('xyz');

SELECT * FROM selection;

-- get all words having any one char between b and g.
SELECT * FROM selection WHERE word REGEXP 'b.g';

-- get all words having any one alphabet between b and g.
SELECT * FROM selection WHERE word REGEXP 'b[a-z]g';

-- get all words having any one letter between b and g (but not alphabet).
SELECT * FROM selection WHERE word NOT REGEXP 'b[a-z]g';

SELECT * FROM selection WHERE word REGEXP 'b[^a-z]g';

-- get all words having any one letter between b and g out of a, i or u
```

```
SELECT * FROM selection WHERE word REGEXP 'b[aiu]g';

-- get all words 'b*g'
SELECT * FROM selection WHERE word REGEXP 'b*g'; -- wrong out
SELECT * FROM selection WHERE word REGEXP 'b\\*g'; -- \\ removes special meaning
of *
SELECT * FROM selection WHERE word REGEXP 'b[*]g';
```

```
CREATE TABLE repetition(word VARCHAR(40));

INSERT INTO repetition VALUES ('ww'), ('wow'), ('woow'), ('wooow'), ('woooow'),
('wooooow'), ('woooooow'), ('wooooooow');

SELECT * FROM repetition;

-- * means 0 or more occurrences of prev char.
SELECT * FROM repetition WHERE word REGEXP 'wo*w';

-- ? means 0 or 1 occurrence of prev char.
SELECT * FROM repetition WHERE word REGEXP 'wo?w';

-- + means 1 or more occurrences of prev char.
SELECT * FROM repetition WHERE word REGEXP 'wo+w';

-- {n} means n occurrences of prev char.
SELECT * FROM repetition WHERE word REGEXP 'wo{4}w';

-- {m,} means more than m occurrences of prev char.
SELECT * FROM repetition WHERE word REGEXP 'wo{4,}w';

-- {m,n} means m to n occurrences of prev char.
SELECT * FROM repetition WHERE word REGEXP 'wo{3,6}w';
```

```
CREATE TABLE people (id INT, name VARCHAR(20), email VARCHAR(40), phone
VARCHAR(14));

-- get all records where phone numbers are valid
--      10 digits -- 9876543210 --> ^[0-9]{10}$
--      0 can be before 10 digit -- 09876543210 --> ^0?[0-9]{10}$
--      +91 can be before 10 digit -- +919876543210 --> ^(\\+91)?[0-9]{10}$ --> ?
check if (+91) group is present/absent
--      0 or +91 can be before 10 digit --> 09876543210 or +919876543210
--                      --> '^(0|\\+91)?[0-9]{10}$'
```