Chapter 3: SQL Statement, Operators and Functions

Analytics Tensor
Mahesh KC
mahesh.kc@analyticstensor.com
https://analyticstensor.com

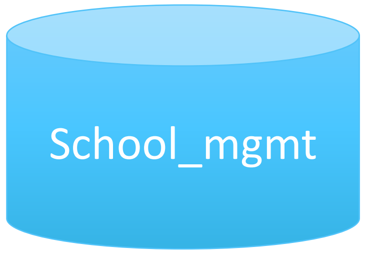
Introduction

- SQL Statement Syntax
- Utility Statement
- Query Clauses
- Filtering
- Operators
  - Assignment Operators
  - Arithmetic Operators
  - Logical Operators
  - Comparison Operators
  - Bitwise Operators
- Operator Precedence
- Functions
- Practical: Install Mysql Database, Creating and Populating Employee Database.
- Wrap-up

SQL Statement Syntax

DDL: Create/Drop Database Syntax

Create is used to create database. Drop is used to drop all tables in database and delete database.



CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] *db_name* [*create_specification*] ...
create_specification: [DEFAULT] CHARACTER SET [=] *charset_name* | [DEFAULT] COLLATE [=] *collation_name* | DEFAULT ENCRYPTION [=] {'Y' | 'N'}

DROP {DATABASE | SCHEMA} [IF EXISTS] *db_name*

Example:
CREATE DATABASE IF NOT EXISTS *school_mgmt;*

Example:
DROP DATABASE IF EXISTS *school_mgmt*;

SQL Statement Syntax

DDL: Create/Drop Table Syntax

Create a table in database. Drop is used to remove tables from database.

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *tbl_name* (
col_name data_type [NOT NULL | NULL
[AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
[COMMENT '*string*']
[*reference_definition*] [*check_constraint_definition*]

Detail Create Table
https://dev.mysql.com/doc/refman/8.0/en/create-table.html

DROP [TEMPORARY] TABLE [IF EXISTS] *tbl_name* [, *tbl_name*]
... [RESTRICT | CASCADE]

student_id	first_name	last_name	dob	add_dt

Example:

```
CREATE TABLE student (
  student_id int(11) NOT NULL,
  first_name varchar(20) NOT NULL,
  last_name varchar(20) NOT NULL,
  dob date NOT NULL,
  add_date date NOT NULL,
  PRIMARY KEY (student_id)
);
```

DROP TABLE IF EXISTS *student*;

SQL Statement Syntax

student_id	first_name	last_name	dob	add_dt
1	John	Doe	1980-01-01	2019-07-04 18:03:22

DML: Insert/Update Syntax:

Insert is used to inserts new rows into an existing table. Update is used to modify rows in a table.

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE] [INTO] *tbl_name* [PARTITION (*partition_name* [, *partition_name*] ...)] [(*col_name* [, *col_name*] ...)] {VALUES | VALUE} (*value_list*) [, (*value_list*)] ...

UPDATE [LOW_PRIORITY] [IGNORE] *table_reference* SET *assignment_list* [WHERE *where_condition*] [ORDER BY ...] [LIMIT *row_count*]

Example:
INSERT TABLE student (
  student_id, first_name, last_name, dob
date, add_date) VALUES (1,'John',
'Doe',1980-01-01, now())
);

UPDATE *student set last_name='Dooe' WHERE student_id=1; -- will update Doe to Dooe*

SQL Statement Syntax

DML: Select Syntax : Select is used to retrieve rows from one or more tables.

SELECT [ALL | DISTINCT | DISTINCTROW] [HIGH_PRIORITY] [STRAIGHT_JOIN] [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT] [SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] *select_expr* [, *select_expr* ...] [FROM *table_references* [PARTITION *partition_list*] [WHERE *where_condition*] [GROUP BY {*col_name* | *expr* | *position*}, ... [WITH ROLLUP]] [HAVING *where_condition*] [WINDOW *window_name* AS (*window_spec*) [, *window_name* AS (*window_spec*)] ...] [ORDER BY {*col_name* | *expr* | *position*} [ASC | DESC], ... [WITH ROLLUP]] [LIMIT {[*offset*,] *row_count* | *row_count* OFFSET *offset*}]

Example:
SELECT student_id, first_name from student ;

SQL Statement Syntax

TCL: Start Transaction, Commit, and Rollback

START TRANSACTION [*transaction_characteristic* [, *transaction_characteristic*] ...] *transaction_characteristic*: { WITH CONSISTENT SNAPSHOT | READ WRITE | READ ONLY } BEGIN [WORK] COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE] ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE] SET autocommit = {0 | 1}

Example:

```
START TRANSACTION;
    SELECT @A:=SUM(salary) FROM
    table1 WHERE type=1; UPDATE table2
    SET summary=@A WHERE type=1;
COMMIT;
```

TCL is used to control transactions.

- START TRANSACTION or BEGIN start a new transaction.
- COMMIT commits the current transaction, making its changes permanent.
- ROLLBACK rolls back the current transaction, canceling its changes.
- SET autocommit disables or enables the default autocommit mode for the current session.

SQL Statement Syntax

Utility Statement: Describe, Explain, Help, Use

Describe and Explain are used to obtain information about table structure or query execution plans. Use statement is used to choose the current database.

Help is used to display online help documentation.

```
DESCRIBE student;
EXPLAIN student;
USE school_mgmt.;
HELP CREATE TABLES;
```

Query Clauses

SELECT statement is composed of several components or clauses. Only select clause is mandatory. Mostly three or more than three clauses are used heavily to query the data.

Clause name	**Purpose**
Select	Determines which columns to include in the query's result set
From	Identifies the tables from which to draw data and how the tables should be joined
Where	Filters out unwanted data
Group by	Used to group rows together by common column values
Having	Filters out unwanted groups
Order by	Sorts the rows of the final result set by one or more columns

Query Clauses (cont.)

```
mysql -u root –p

Show databases ; -- display list of databases
Use employees ; -- choose employees database
Show tables ; -- show all the tables in employees database
Describe employees ; -- show employees table structure
Show create table employees ; -- display structure to create employees table.
```

SELECT

SELECT clause determines all possible columns to be included in the query's resultset.

- * mean choose all columns.
- Select clause also allow to add several functionality:
- Literals, such as numbers or strings
- Expressions, allow operator to manipulate with columns values
- Built-in functions calls, allow all the functions calls
- User-defined functions calls, allow UDF functions calls

select dept_no, dept_name from departments ; -- *display dept_no, dept_name from departments table*

select * from departments ; -- select all columns from departments

select 'Analytics Tensor' as company,emp_no, concat_ws(',first_name,last_name) as fullname, month(birth_date) birth_month, birth_date from employees limit 10 ;

select version(), user(), database(),now() ; -- execute built-in or simple expression

select distinct dept_no from dept_emp ; -- display unique dept number from dept_emp table

FROM

FROM clause contains list of one or more tables. It can contains:

Permanent tables, stored table.

Temporary tables, rows from subquery

Virtual tables, view

```
select dept_no, dept_name from departments ;  --using permanent table
select * from (select emp_no, first_name,last_name from employees) e limit 10 ; -- using temporary table
select * from current_dept_emp limit 10 ; --using view
```

WHERE

WHERE clause is used to filter unwanted rows from the resultset.

Where clause contains one or multiple filter conditions. It uses operator such as and, or and not.

```
select * from employees where first_name like 'M%' limit 10 ; -- display all employees whose first_name starts with M.
select *,datediff(now(),hire_date) as hire_dt from employees where first_name like 'S%' and gender='F' order by hire_dt desc limit 10 ;
-- select female employee whose name start with S  order by hire_dt
```

GROUP BY and HAVING

Group By and Having are used in data aggregation. Group by is used to group the data by specified column values. Having is used to filter group data similar to where clause.

```
select emp_no, count(*) total_chg from salaries group by emp_no having total_chg > 10 order by total_chg desc
-- count total salaries change of each employee where changes is more than 10 time and sort by max changes.
```

ORDER BY

ORDER BY is used to sort the data/resultset based on column data. It can contains one or more columns to sort the data. The records are sorted by column order if it contains more columns in order by clause. The record are sorted either ASC (default) or DESC. It can also use column by position instead of column names.

```
select * from current_dept_emp order by dept_no limit 10 ;
-- order data by dept_no
Select* from employees order by 4, 3; -- sort by last_name, first_name.
```

Operators

- Assignment Operators
- Arithmetic Operators
- Bitwise Operators
- Logical Operators
- Comparison Operators

Assignment Operators

Name	Description
=	Assign a value. It is a part of a SET clause in an UPDATE statement)
:=	Assign a value

Arithmetic Operators

Name	**Description**
DIV	Integer division
/	Division operator
-	Minus operator
%, MOD	Modulo operator
+	Addition operator
*	Multiplication operator
-	Change the sign of the argument

Bitwise Operators

Name	**Description**
&	Bitwise AND
~	Bitwise inversion
|	Bitwise OR
^	Bitwise XOR
<<	Left shift
>>	Right shift

Logical Operators

Name	**Description**
AND	Logical AND
NOT	Negates value
OR	Logical OR
XOR	Logical XOR

Comparison Operators

Name	**Description**
=	Equal operator
<->	NULL-safe equal to operator
>	Greater than operator
>=	Greater than or equal operator
<	Less than operator
<=	Less than or equal operator
!=, <>	Not equal operator

Operators Precedence

!
- (unary minus), ~ (unary bit inversion)
^
*, /, DIV, %, MOD
-, +
<<, >>
&
|
= (comparison), <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
BETWEEN, CASE, WHEN, THEN, ELSE
NOT
AND, &&
XOR
OR, ||
= (assignment), :=

Functions

- Numeric Functions
- String Functions
- Date and Time Functions
- Control Flow Functions
- Cast Functions
- Encryption and Compression Functions
- Aggregate/Grouping Functions (Important)
- Window Functions
- Information Functions
- JSON Functions

Wrap-up

- Important points.
- Q & A.

Assignment-4

Choose top 10 important function for each group at slide 23. And write one SQL statement using employee database. You can choose and tables. If you are familiar with JOIN then you can utilize those functionality. Output file format should be JSON file. All other file will be ignore and graded with 0.

Optional: Extra credit 80 point if you create all the 100 SQL statement using at least join or subquery.

Example:

```
{
        ”name”:”concat_ws(string separator, string_1, string_2)”
        “usage”:”concat two string with string separator”
        “string_separator”:’ ‘
        “string_1”:”first_name
        “string_2”:”lastr_name”
        “database”:”employee”
        “sql_statement”:”select concat_ws(‘ ‘ , first_name, last_name from employee as
                ’employee_fullname’”)
}
```

References

https://dev.mysql.com/doc/refman/8.0/en/