SQL

(STRUCTURED QUERY LANGUAGE)

USING

MySQL

RAJEEV SRIVASTAVA (RAJEEVS@CDAC.IN)

SQL: STRUCTURED QUERY LANGUAGE

- Stands for **“Structured Query Language”**
- Also pronounced as “**SEQUEL**” (Structured English QUEry Language)
- Originally developed at **IBM** in the 1970s by **Donald Chamberlin** and **Raymond Boyce**
- Standard access mechanism to every **RDBMS**.
- Case-Insensitive
- 4th Generation Language (Instructions for WHAT to do)
- Standard Based – ANSI / ISO
- First SQL Standard Published in 1986 (**SQL:86**) by ANSI
- Latest is **SQL:2016** OR **ISO/IEC 9075:2016**

COMPONENTS (CATEGORIES) OF SQL STATEMENTS

- **DDL** : Data Definition Language(CREATE/ALTER/DROP/TRUNCATE)
- **DML**: Data Manipulation Language(INSERT/UPDATE/DELETE)
- **DCL** : Data Control Language (GRANT/REVOKE)
- **DTL** : Data Transaction Language OR
  TCL : Transaction Control Language (COMMIT/SAVEPOINT/ROLLBACK)
- **DRL** : Data Retrieval Language OR
  DQL : Data Query Language (SELECT)

CREATING A TABLE

- Data in a Relational database is stored in the form of tables.
- The table is a collection of related data entries and it consists of columns and rows.

```
CREATE TABLE <tableName>  (
 <columnName>   <dataType>,
 <columnName>   <dataType>
);
```

- SQL statements ends with a Semicolon (;)

Find out Restrictions on Table and Column names in MySQL

Max Size of Table in MySQL

Max Number of Columns in a Table in MySQL

CONSIDERATIONS FOR CREATING TABLE

- Points to be considered before creating a table -
  - What are the **Attributes** (columns/fields) of the tuples(records/rows) to be stored?
  - What are the **Data Types** of the attributes? Should varchar be used instead of char?
  - Which column(s) build the **Primary Key**?
  - What column(s) need to added as **Foreign Keys?**
  - Which column(s) do (not) allow **NULL** values?
  - Which column(s) will have **UNIQUE** values ie. do (not) allow duplicates?
  - Are there **DEFAULT** values for certain columns?

MYSQL: DATA TYPES

Explore different Data Types available in MySQL with their uses.

- Data Type defines what kind of values can be stored in a column.
- Data Type also defines the way data will be stored in the system and the space required in disk.
- Data Type also impact database performance.
- Ex- Char, Varchar, Text, Integer, Float, Double, Date, Timestamp, Enum, Blob etc.
- More on SQL Datatypes : https://www.w3schools.com/sql/sql_datatypes.asp

INSERT

- Used to insert data into a table
- Insert command always inserts values as new row –

```
INSERT INTO <tableName> VALUES (<val1>, <val2>);
```

- Insert data into only specific columns of a table -

```
INSERT INTO <tableName> (<col1>) VALUES (<val1>);
```

- Define an insertion order -

```
INSERT INTO <tableName> (<col2>, <col1>) VALUES (<val2>,
<val1>);
```

- Missing attribute → NULL.
- May drop attribute names if give them in order

NULL VALUE

- When you do not insert data into a column of a table for a specific row, then by default a NULL value will be inserted into that column by the database.

```
INSERT INTO dept (deptno, deptname) VALUES (40, ‘BIOM’);
```

- NULL value does not occupy space in memory
- NULL value is independent of data type
- A NULL value is not a zero (0) OR an empty string (‘ ’), rather it represents an **Unknown** or **Not Applicable** value.

SELECT

- Used to Retrieve/Fetch information from the database.

```
SELECT <colName> FROM <tableName> [WHERE <condition>];
SELECT <col1>, <col2>  FROM <tableName> [WHERE
<condition>];
```

- An asterisk symbol (*) Represents all columns/attributes.

```
SELECT *  FROM <tableName> [WHERE <condition>];
```

COMBINING MORE THAT ONE CONDITIONS (AND/OR), NOT

- More than one conditions may be specified in WHERE clause to fetch the data matching multiple criteria -

```
SELECT <colName> FROM <tableName> [WHERE <condition1>
[AND|OR WHERE <condition2>]…];
```

- AND – will match both the conditions
- OR – will match either of the conditions
- NOT – will display not unmatching records

```
SELECT <colName> FROM <tableName> WHERE NOT (<condition>);
```

SELECTION & PROJECTION

- **SELECTION** (σ) – limiting rows (by using WHERE clause)

```
SELECT * FROM <table name> WHERE <col1> = <val1> ;
```

- **PROJECTION** (π) – limiting columns (by using SELECT clause)

```
SELECT <col1>, <col2> FROM <table name>;
```

- SELECTION & PROJECTION – limiting rows and columns selection (by using SELECT and WHERE clauses together)

```
SELECT <col1>, <col2> FROM <table name> WHERE <col1> =
<val1> ;
```

ALTER TABLE

- To modify/change an existing column

```
ALTER TABLE <tableName> MODIFY COLUMN <col> <newDataType>;
```

- To add new column

```
ALTER TABLE <tableName> ADD COLUMN <col>  <dataType>;
```

- To rename an existing column

```
ALTER TABLE <tableName> RENAME COLUMN <col> TO
<newColumnName>;
```

- To drop/remove an existing column

```
ALTER TABLE <tableName> DROP COLUMN <col>;
```

ALTER TABLE - CONSTRAINTS

- To add a Primary Key constraint in existing table

```
ALTER TABLE <tableName> ADD PRIMARY KEY (<columnName>);
```

- To add a Foreign Key constraint in existing table

```
ALTER TABLE <tableName> ADD FOREIGN KEY (<columnName>)
REFERENCES <refTableName> (<refColumnName>);
```

- To add a other constraints

```
ALTER TABLE <tableName> MODIFY <columnName> <dataType>
NOTNULL;
```

- Dropping a constraint

```
ALTER TABLE <tableName> DROP PRIMARY KEY;
ALTER TABLE <tableName> DROP CONSTRAINT <constraintName>;
```

UPDATE

- This command inserts or modifies values in the cells in existing rows
- This command can also be used for deleting values from a cell of a table without the need for deleting a row or a column

- Syntax

```
UPDATE <table name> SET <col name> = <new value> [WHERE
<condition>];
```

DELETE

- This command is used for deleting specific or all the rows from a table

- Syntax

```
DELETE FROM <table name> [WHERE <condition>];
```

TRUNCATE COMMAND

- This command can also be used for deleting all the rows from a table
- Syntax

```
TRUNCATE TABLE <table name>;
```

- Truncate command cannot be rolled back because it is a AUTO COMMIT operation, i.e. changes committed cannot be rolled back. But DELETE is not a AUTO COMMIT operation. Hence DELETE can be ROLLED BACK.
- Truncate is a DDL Command.

DROP COMMAND

- DROP command can be used for permanently deleting database objects like table, view, function etc. from a database
- Deletes the entire object definition
- Can’t be rolled back

- Syntax

```
DROP TABLE <table name>;
```

CONSTRAINTS

- PRIMARY KEY
- FOREIGN KEY
- UNIQUE
- NOT NULL
- DEFAULT
- CHECK (NOT SUPPORTED BY MySQL)

RELATIONAL OPERATORS

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A=A) is true (A = B) is not true.
!= <>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true

BETWEEN … AND… OPERATOR

- This operator is used as a replacement for relational (>, <) and logical operators (AND, OR)
- In this operator lower and upper values are inclusive if they are of number or date types
- The lower limit must be <= upper limit

```
SELECT * FROM emp WHERE sal BETWEEN 10000 AND 20000;
```

IS NULL OPERATORS

- Used for testing for the existence of NULL values in any column
- Ex. Display the details of those employees who are not having any job

```
SELECT * FROM emp WHERE job IS NULL:
```

- IS NOT NULL

```
SELECT * FROM emp WHERE job IS NOT NULL:
```

- A NULL value cannot be compared. Hence you cannot use relational operators for comparing NULL value with a column
- Therefore IS operator has to be used for the purpose

IN/NOT IN OPERATOR

- This operator is used to compare multiple values within a single column
- Works with all data types
- Values supplied must be of the same type and must belong to only 1 column
- This operator is a replacement for multiple OR operators

```
SELECT * FROM emp WHERE job IN (‘PE', ‘TO‘, ‘SE’);
```

- NOT IN operator is used to exclude multiple matching values -

```
SELECT * FROM emp WHERE job NOT IN (‘PE’, ‘TO’);
```

LIKE OPERATOR - % AND _

- This operator is used for comparing characters/numbers in a value from a specific position
  - % ignores variable number of characters
  - _ ignores only 1 char
- Display the details of those emps whose name is starting with ‘r’

```
SELECT * FROM emp WHERE ename LIKE ‘r%’;
```

- Display the details of those emps who have ‘h’ as second character in their name –

```
SELECT * FROM emp WHERE ename LIKE ‘_h%’;
```

- In MySQL, Case of character values is ignored by Like operator.

DISTINCT - ELIMINATING DUPLICATES

- DISTINCT command is used to select only unique values from a column. i.e. only single occurrence of a value will be returned.

```
SELECT DISTINCT job FROM emp;
SELECT DISTINCT loc FROM dept;
SELECT DISTINCT job, sal FROM emp;
```

FUNCTIONS

- Function is a Sub Program which performs a specific task
- Every function returns only 1 value
- Functions in MySQL database can be used or defined for -
  - Performing arithmetic calculations which are not possible/easy using arithmetic operators
  - Formatting text/numbers/dates
  - Type casting i.e. converting one type of data into another
  - To fetch information from system schema. Eg. VERSION()

FUNCTION – TYPES AND USES

- Types of Functions
  - System defined functions
  - User defined functions
- Syntax –

```
SELECT <function name(args)> [FROM <tableName>];
```

FUNCTIONS – SYSTEM DEFINED

- Numeric functions
- String functions
- Date and Time functions
- Conversion functions
- **Aggregate functions**
- More Functions -
  - https://www.w3schools.com/sql/sql_ref_mysql.asp

FEW FUNCTIONS

- Aggregate Functions

COUNT()	: Gives count of occurrences; Works on All data types.
AVG()	: Average. Works only on Numeric values
SUM()	: Gives total/sum. Works only on Numeric values
MAX()	: Maximum Value. All data types
MIN()	: Minimum Value. All data types

- Concatenation Function - Used to combine/merge two or more values

  CONCAT (<col/value1>, <col/value2>, …)

SOME IMPORTANT FUNCTIONS

Check syntax of these functions and use these in SQL queries

- FORMAT
- LEFT
- LENGTH
- LOWER
- LOCATE
- LPAD
- LTRIM
- REPLACE
- REVERSE
- RIGHT
- RPAD
- RTRIM
- SUBSTR
- TRIM
- UCASE
- UPPER
- ABC
- CEIL
- EXP
- FLOOR
- MOD
- ROUND
- SQRT
- DATE
- DAY
- SECOND
- MINUTE
- HOUR
- DAY
- MONTH
- YEAR
- NOW
- SYSDATE
- EXTRACT
- LAST_DAY
- DATE_FORMAT

REAL TIME USE OF FUNCTIONS

- Print all department names in Upper Case -

```
SELECT UPPER(dname) FROM dept;
```

- Find out maximum salary from Emp table -

```
SELECT MAX(sal) FROM emp;
```

- How much is the average salary of employees?. Also find out total amount paid as salary.

```
SELECT SUM(sal), AVG(SAL) FROM emp;
```

ORDER BY, GROUP BY AND HAVING

- ORDER BY (if used, ORDER BY must by last clause of a query, except when you have used LIMIT Clause)

```
SELECT * FROM emp ORDER BY sal;
SELECT * FROM emp ORDER BY sal DESC;
SELECT * FROM emp ORDER BY sal DESC, ename;
```

- GROUP BY

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

- GROUP BY….HAVING

```
SELECT deptno, SUM(sal) FROM emp GROUP BY deptno HAVING
SUM(sal) < 50000;
```

LIMITING NO OF RECORDS

- LIMIT clause is used to limit number of records returned by a SELECT query.
- Its not part of SQL standard, works in MySQL, and few other RDBMS.
- Unexceptionally, must be the last clause of a query.

```
SELECT * FROM emp LIMIT 2;
SELECT * FROM emp ORDER BY sal DESC LIMIT 3;
```

COPYING TABLE/DATA

- A copy of the table (with, without or selected data)can be created using SELECT command

```
CREATE table dept_copy AS SELECT * FROM dept;
CREATE table emp_pe AS SELECT * FROM emp WHERE job = ‘PE’;
CREATE table emp_new AS SELECT * FROM emp WHERE 1=2;
```

- To copy only data of a table to another table

```
INSERT INTO emp_seng SELECT * FROM emp WHERE deptno = 10;
INSERT INTO emp_seng SELECT * FROM emp WHERE deptno =
(SELECT deptno FROM dept WHERE dname = ‘SENG’);
```

COMMIT, SAVEPOINT & ROLLBACK…

- By default autocommit parameter is ON in MySQL and can be reconfigured by MySQL Administrator. A user can set autocommit parameter to ON & OFF for herself using SET command
- To check the status of autocommit for the user –

```
SHOW LOCAL VARIABLES LIKE ‘autocommit’;
```

- To set the autocommit ON (1) /OFF (0)

```
SET autocommit=[0|1]
SET autocommit=[ON|OFF]
```

COMMIT, SAVEPOINT & ROLLBACK

- Savepoint and Rollback commands will be effective only when autocommit is OFF;
- System (MySQL) autocommits all uncommitted operations before executing any DDL command.

```
COMMIT;
SAVEPOINT <variable name>;
ROLLBACK [ TO <variable name>];
```

- COMMIT to a particular SAVEPOINT is not supported by MySQL.

JOINS

- Joins is a technique of retrieving data from multiple tables

- Display the ename AND dname from emp and dept tables

```
SELECT ename, dname FROM emp, dept  WHERE emp.deptno =
dept.deptno;
```

- Display the ename and dname of those emps whose sal is > 20000

```
SELECT ename, dname, sal FROM emp, dept WHERE emp.deptno =
dept.deptno AND sal > 20000;
```

JOIN TYPES : SQL STANDARD SYNTAX

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: (NOT Supported by MySQL) Return all records when there is a match in either left or right table

```
SELECT <columnNames> FROM <tableName1>
LEFT|RIGHT OUTER JOIN <tableName2> ON
<tableName1>.<columnName> = <tableName2>.<columnName>;
```

INNER JOIN: EXAMPLES

```
SELECT empno, ename, sal, dname
FROM dept JOIN emp
ON dept.deptno = emp.deptno;

SELECT *
FROM dept JOIN emp
ON dept.deptno = emp.deptno;
```

LEFT OUTER JOIN: EXAMPLE

```
SELECT empno, ename, sal, dname
FROM dept LEFT OUTER JOIN emp
ON dept.deptno = emp.deptno;

SELECT *
FROM dept LEFT OUTER JOIN emp
ON dept.deptno = emp.deptno;
```

RIGHT OUTER JOIN: EXAMPLE

```
SELECT empno, ename, sal, dname
FROM dept RIGHT OUTER JOIN emp
ON dept.deptno = emp.deptno;

SELECT *
FROM dept RIGHT OUTER JOIN emp
ON dept.deptno = emp.deptno;
```

SUBQUERIES

- It is a query within another query

- Display the details of those employees who are getting a sal > Aditya

```
SELECT * FROM emp WHERE sal > (SELECT sal FROM emp WHERE
ename = ‘Aditya’;
```

- Display details of all employees who work in department ‘SENG’

```
SELECT * FROM emp WHERE DEPTNO = (SELECT deptno FROM dept
WHERE dname = ‘SENG’);
```

MORE SUBQUERIES EXAMPLES

- Display the maximum salary from every department and also the name of that employee who is getting the maximum Salary.

```
SELECT ename, sal, deptno FROM emp WHERE (deptno, sal) IN
(SELECT deptno, max(sal) FROM emp GROUP BY deptno);
```

SET OPERATIONS: UNION & UNION ALL

- Set operations treat the tables as sets and are the usual set operators of union, intersection, and difference
- MySQL Supports only UNION [ALL] operators

```
SELECT * FROM <table name 1>
UNION [ALL]
SELECT * FROM <table name 2>;
```

- SET operations must fulfil following two conditions –
  - Number of columns in the SELECT clause of all queries must be same.
  - Data types of respective columns in all queries must be same or compatible.

VIEWS

- Views are ‘Virtual Tables’.
- Views does not store data but fetches the data from underlying tables[s] dynamically at runtime.

```
CREATE VIEW <view name> AS <Select Query> ;
```

- All DML operations can be performed on a view and it affects underlying table.