Forthbridge Analytics NG

SQL Basics

Femi Ayodele

Data Analyst || BI Developer ||Analytics Engineer

Email: femiayodele@yahoo.com

LinkedIn: Femi Ayodele (MCT) | LinkedIn

Tel: +2349135067747

What is SQL?

- SQL stands for Structured Query Language. It is used for storing and managing data in Relational Database Management System (RDBMS).
- It is a standard language for Relational Database System. It enables a user to relational databases and tables.
- All the RDBMS like **MySQL, Informix, Oracle, MS Access and SQL Server** use SQL as their standard database language.
- SQL allows users to query the database in a number of ways, using English-like statements.

What are the SQL?

SQL follows the following rules:

- Structure query language is not case sensitive. Generally, keywords of SQL are written in uppercase.
- Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text lines.
- Using the SQL statements, you can perform most of the actions in a database.
- SQL depends on tuple relational calculus and relational algebra.

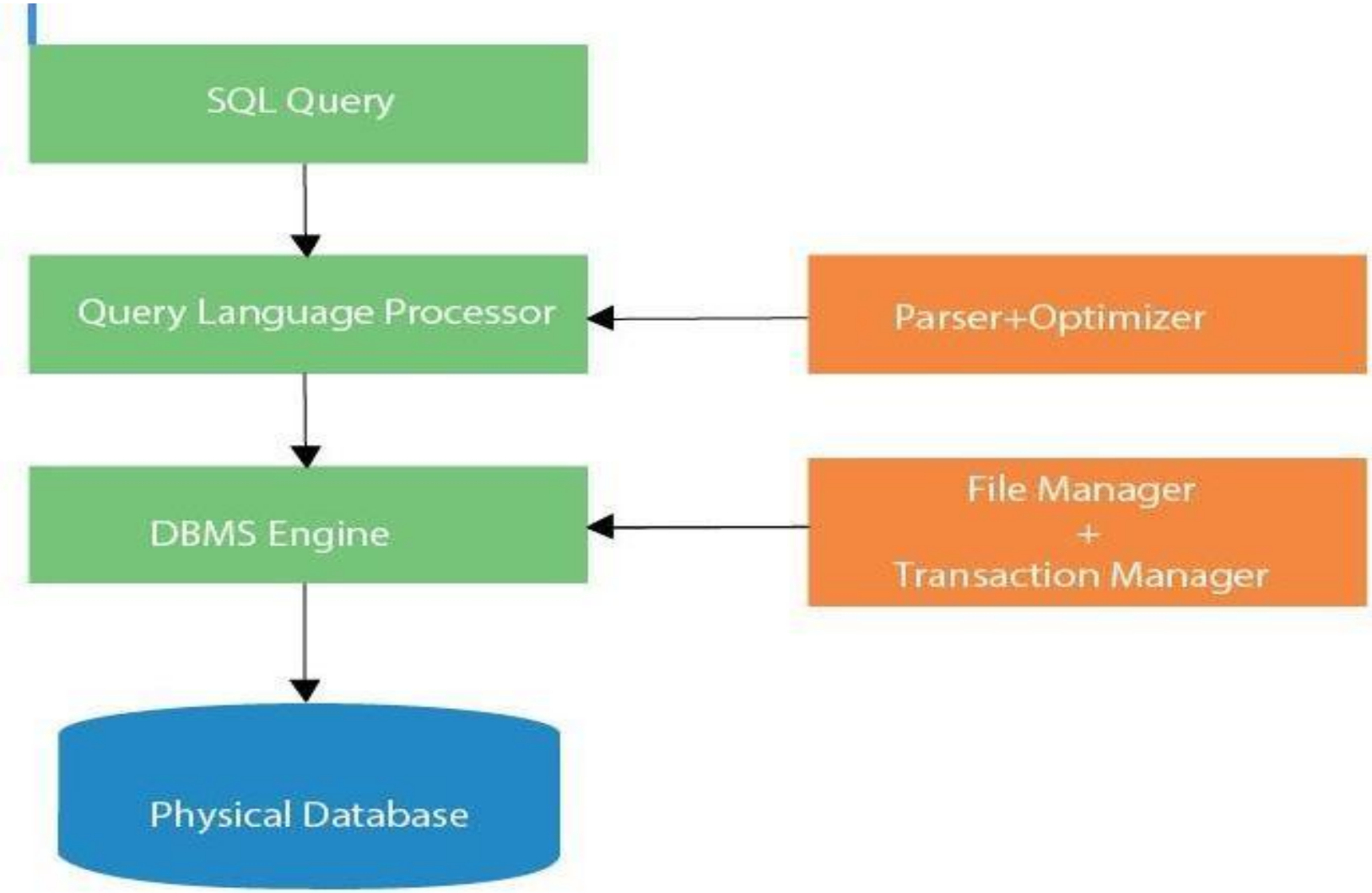
What is SQL Process?

- When an SQL command is executing for any RDBMS, then the system figure out the best way to carry out the request and the SQL engine determines that how to interpret the task.

- In the process, various components are included. These components can be optimization Engine, Query engine, Query dispatcher, classic, etc.

- All the non-SQL queries are handled by the classic query engine, but SQL query engine won't handle logical files.

What is SQL Process?

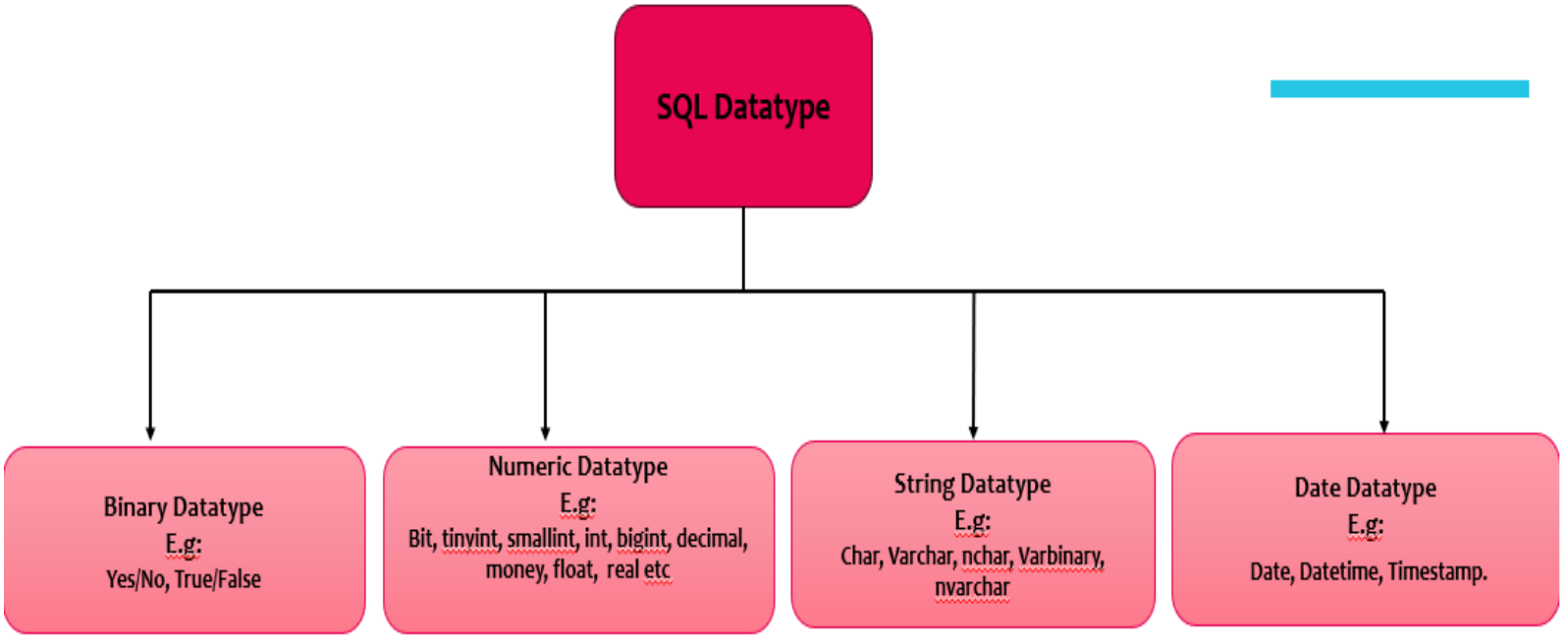


What is Advantages of SQL?

- High speed
- No coding needed
- Well defined standards
- Portability
- Interactive language
- Multiple data view

What is SQL Datatype?

- SQL Datatype is used to define the values that a column can contain.
- Every column is required to have a name and data type in the database table.



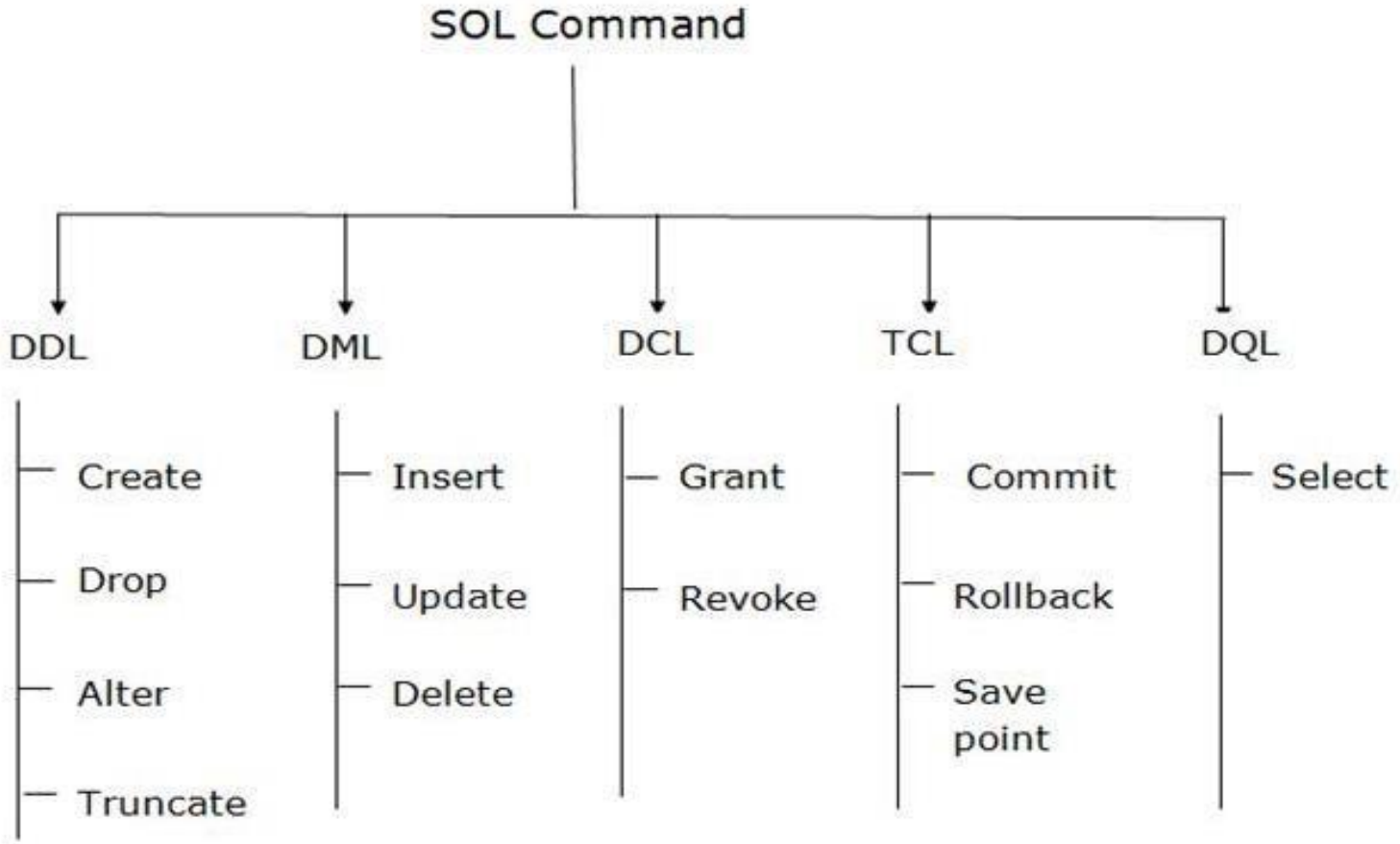
SQL Commands

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.

- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

Types of SQL Commands

- There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.

- All DDL commands are auto-committed, meaning it permanently saves all the database changes.
- Here are some commands that come under DDL:
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE

Data Definition Language (DDL)- CREATE

CREATE It is used to create a new table in the database.

Syntax:

REATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES [, .]);

Example:

CREATE TABLE EMPLOYEE (Name VARCHAR2(20), Email VARCHA R2(100), DOB DATE);

Data Definition Language (DDL)- Drop

Drop: It is used to delete both the structure and record stored in the table.

Syntax:

DROP TABLE ;

Example:

DROP TABLE EMPLOYEE;

Data Definition Language (DDL)- **ALTER**

ALTER: It is used to alter the structure of the database. Thischange could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

ALTER TABLE table_name ADD column_name COLUMN-definition;

ALTER TABLE MODIFY(COLUMN DEFINITION...);

Example:

ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));

ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

Data Definition Language (DDL)- **TRUNCATE**

TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Syntax:

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE TABLE EMPLOYEE;
```

Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of CHANGES in the database.

- The command of **DML is not auto-committed** that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- **INSERT**
- **UPDATE**
- **DELETE**

Data Manipulation Language - **INSERT**

INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

```
INSERT INTO TABLE_NAME    (col1, col2, col3,.... col N)
VALUES (value1, value2, value3,..... valueN);
```

OR

```
INSERT INTO TABLE_NAME    VALUES (value1, value2, value3, ..... valueN);
```

Example:

```
INSERT INTO XYZ (Author, Subject) VALUES ("Sonoo", "DBMS");
```

Data Manipulation Language - **UPDATE**

Update: This command is used to **update or modify** the value of a column in the table.

Syntax:

UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]

Example:

```
UPDATE students
SET User_Name = 'Sonoo'
WHERE Student_Id = '3'
```

Data Control Language

DCL commands are used to GRANT and TAKE BACK authority from any database user.

Here are some commands that come under DCL:

- Grant
- Revoke

Data Control Language - **Grant**

GRANT: It is used to give user access privileges to a database.

Example:

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOT
HER_USER;
```

REVOKE: It is used to take back permissions from the user.

Example:

```
REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

Transaction Control Language - COMMIT

Commit: Commit command is used to save all the transactions to the database.

Syntex:

```
COMMIT;
```

Example:

```
DELETE FROM CUSTOMERS
WHERE AGE = 25;
COMMIT;
```

Transaction Control Language - **Rollback**

Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntex:

```
ROLLBACK;
```

Example:

```
DELETE FROM CUSTOMERS
WHERE AGE = 25;
ROLLBACK;
```

SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntex:

```
SAVEPOINT SAVEPOINT_NAME;
```

Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

SELECT

a. **SELECT:** This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

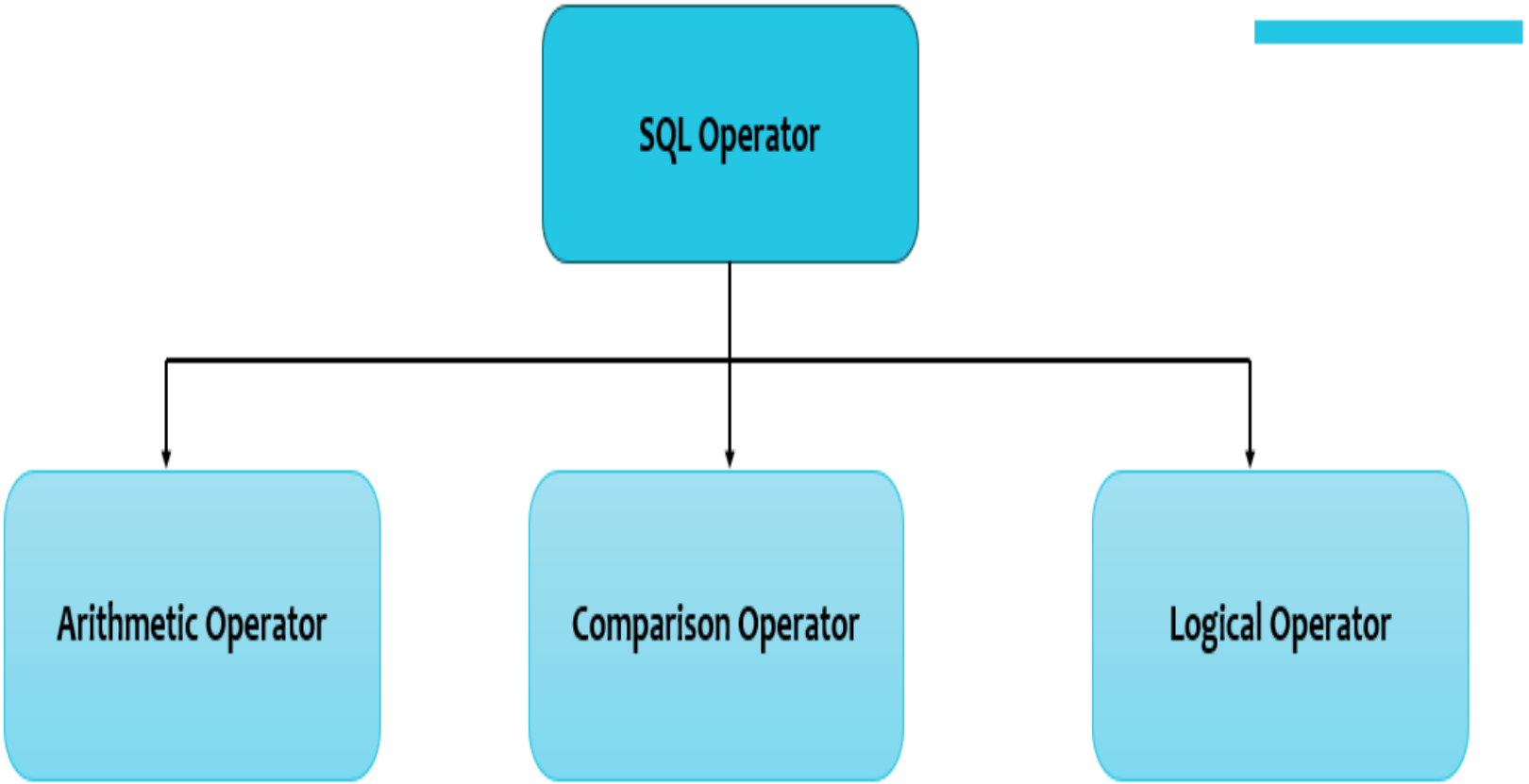
Syntax:

```
SELECT expressions    FROM TABLES    WHERE conditions;
```

Example:

```
SELECT emp_name  FROM employee  WHERE age > 20;
```

SQL Operator



SQL Comparison Operators:

Operator	Description
+	It adds the value of both operands.
-	It is used to subtract the right-hand operand from the left-hand operand.
*	It is used to multiply the value of both operands.
/	It is used to divide the left-hand operand by the right-hand operand.
%	It is used to divide the left-hand operand by the right-hand operand and returns reminder.

SQL Arithmetic Operators

Operator	Description
=	It checks if two operands values are equal or not, if the values are queal then condition becomes true.
!=	It checks if two operands values are equal or not, if values are not equal, then condition becomes true.
<>	It checks if two operands values are equal or not, if values are not equal then condition becomes true.
>	It checks if the left operand value is greater than right operand value, if yes then condition becomes true.
<	It checks if the left operand value is less than right operand value, if yes then condition becomes true.
>=	It checks if the left operand value is greater than or equal to theright operand value, if yes then condition becomes true.

SQL Arithmetic Operators

Operator	Description
<=	It checks if the left operand value is less than or equal to the right operand value, if yes then condition becomes true.
!<	It checks if the left operand value is not less than the right operand value, if yes then condition becomes true.
!>	It checks if the left operand value is not greater than the right operand value, if yes then condition becomes true.

SQL Logical Operators

Operator	Description
All	It compares a value to all values in another value set.
AND	It allows the existence of multiple conditions in an SQL statement.
ANY	It compares the values in the list according to the condition.
Between	It is used to search for values that are within a set of values.
IN	It compares a value to that specified list value.
NOT	It reverses the meaning of any logical operator.
OR	It combines multiple conditions in SQL statements.
EXIST	It is used to search for the presence of a row in a specified table.
LIKE	It compares a value to similar values using wildcard operator.

Example:

```
SQL> CREATE TABLE EMPLOYEE (
EMP_ID INT                    NOT NULL,
EMP_NAME VARCHAR (25) NOT NULL,
PHONE_NO INT                  NOT NULL,
ADDRESS CHAR (30),
PRIMARY KEY (ID)
);
```

- **DESC EMPLOYEE;**
- DELETE FROM table_name WHERE condition
- DROP TABLE "table_name";
- SELECT * FROM table_name;
- INSERT INTO TABLE_NAME VALUES (value1, value2, value 3, ... Value N);
- INSERT INTO TABLE_NAME[(col1, col2, col3,.... col N)] VALUES (value1, value2, valu e 3,..... Value N);
- UPDATE table_name SET column_name = value WHERE condition;

Example:

- UPDATE table_name SET column_name = value1, column_name2 = value WHERE condition;
- DELETE FROM table_name WHERE some_condition;

Views in SQL

- Views in SQL are considered as a virtual table. A view also contains rows and columns.
- To create the view, we can select the fields from one or more tables present in the database.
- A view can either have specific rows based on certain condition or all the rows of a table.

Creating view

A view can be created using the **CREATE VIEW** statement. We can create a view from a single table or multiple tables.

Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

Creating View from a single table

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM Student_Details
WHERE STU_ID < 4;
```

Creating View from multiple tables

View from multiple tables can be created by simply include multiple tables in the SELECT statement.

In the given example, a view is created named MarksView from two tables Student_Detail and Student_Marks.

```
CREATE VIEW MarksView AS
SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Student_Marks.MARKS
FROM Student_Detail, Student_Mark
WHERE Student_Detail.NAME = Student_Marks.NAME;

SELECT * FROM MarksView;

DROP VIEW view_name;
```

SQL Index

- Indexes are special lookup tables. It is used to retrieve data from the database very fast.
- An Index is used to speed up select queries and where clauses. But it shows down the data input with insert and update statements. Indexes can be created or dropped without affecting the data.
- An index in a database is just like an index in the back of a book.

Create Index statement

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

Unique Index statement

Syntax

CREATE UNIQUE INDEX index_name ON table_name (column1, colum n2, ...);

Example

CREATE UNIQUE INDEX websites_idx ON websites (site_name);

Drop Index Statement

Syntax

DROP INDEX index_name;

Example

DROP INDEX websites_idx;

SQL Sub Query

A Subquery is a query within another SQL query and embedded within the WHERE clause.

Important Rule:

- A subquery can be placed in a number of SQL clauses like WHERE clause, FROM clause, HAVING clause.
- You can use Subquery with SELECT, UPDATE, INSERT, DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.
- A subquery is a query within another query. The outer query is known as the main query, and the inner query is known as a subquery.
- Subqueries are on the right side of the comparison operator.
- A subquery is enclosed in parentheses.
- In the Subquery, ORDER BY command cannot be used. But **GROUP BY command can be used** to perform the same function as ORDER BY command.

Subqueries with the Select Statement

SQL subqueries are most frequently used with the Select statement.

Syntax:

```
SELECT column_name
FROM table_name
WHERE column_name expression operator
( SELECT column_name from table_name WHERE ... );
```

Example:

```
SELECT *
   FROM EMPLOYEE
   WHERE ID IN (SELECT ID
   FROM EMPLOYEE
   WHERE SALARY > 4500);
```

Subqueries with the INSERT Statement

- SQL subquery can also be used with the Insert statement. In the insert statement, data returned from the subquery is used to insert into another table.
- In the subquery, the selected data can be modified with any of the character, date functions.

Syntax:

```
INSERT INTO table_name (column1, column2, column3 ....)
    SELECT *  FROM table_name    WHERE VALUE OPERATOR
```

Example:

```
INSERT INTO EMPLOYEE_BKP
    SELECT * FROM EMPLOYEE
    WHERE ID IN (SELECT ID
    FROM EMPLOYEE);
```

Subqueries with the UPDATE Statement

The subquery of SQL can be used in conjunction with the Update statement. When a subquery is used with the Update statement, then either single or multiple columns in a table can beupdated.

Syntax:

```
UPDATE table   SET column_name = new_value WHERE VALUE OPERATOR
    (SELECT COLUMN_NAME   FROM TABLE_NAME WHERE condition);
```

Example:

Let's assume we have an EMPLOYEE_BKP table available which is backup of EMPLOYEE table. The given example updates the SALARY by .25 times in the EMPLOYEE table for all employee whose AGE is greater than or equal to 29.

```
UPDATE EMPLOYEE
    SET SALARY = SALARY * 0.25
        WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP
            WHERE AGE >= 29);
```

Subqueries with the DELETE Statement

The subquery of SQL can be used in conjunction with the Delete statement just like any other statements mentioned above.

Syntax:

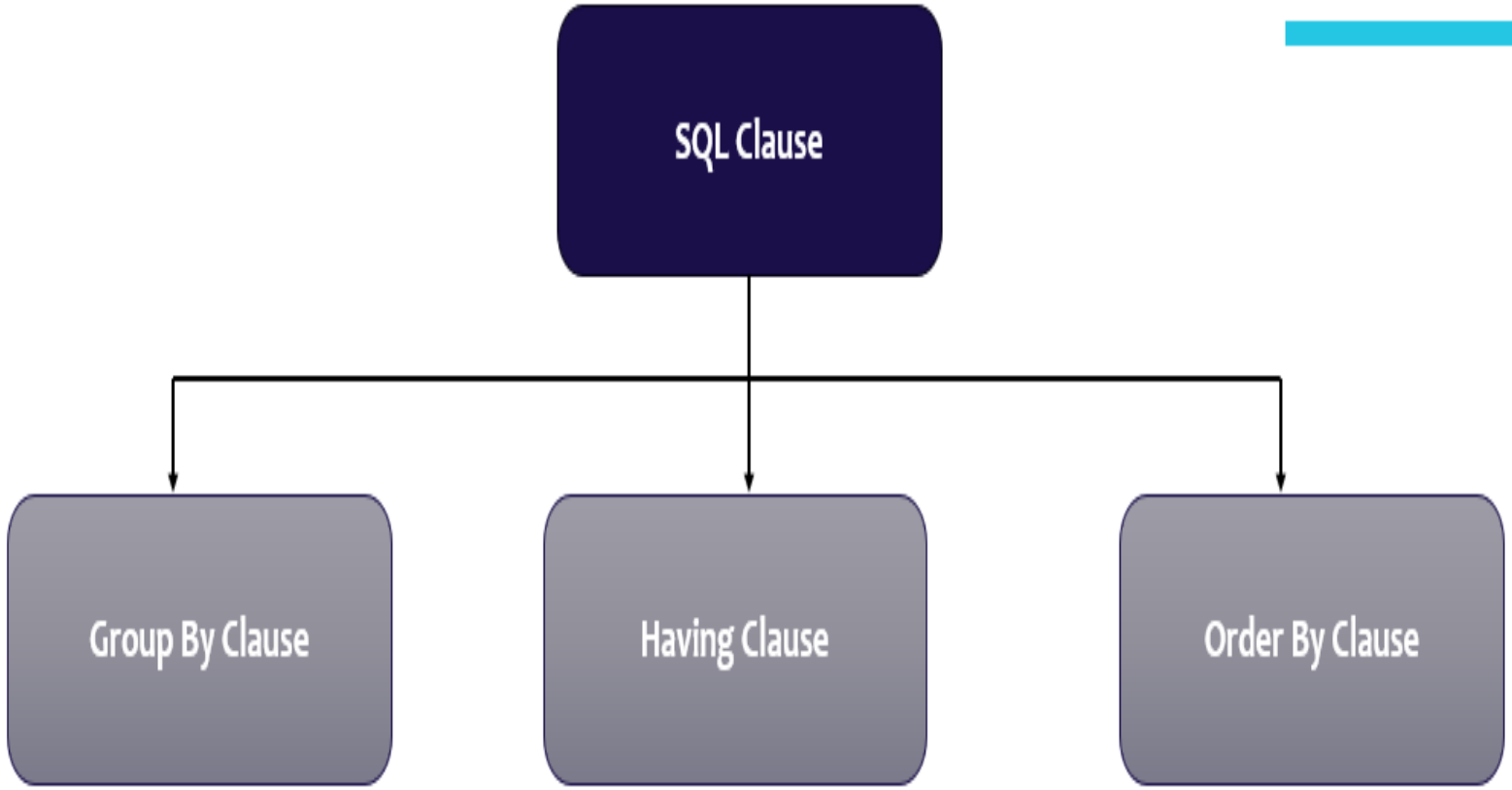
```
DELETE FROM TABLE_NAME   WHERE VALUE OPERATOR
        (SELECT COLUMN_NAME  FROM TABLE_NAME   WHERE condition);
```

Example:

Let's assume we have an EMPLOYEE_BKP table available which is backup of EMPLOYEE table. The given example deletes the records from the EMPLOYEE table for all EMPLOYEE whose AGE is greater than or equal to 29.

```
DELETE FROM EMPLOYEE
  WHERE AGE IN (SELECT AGE FROM EMPLOYEE_BKP
    WHERE AGE >= 29 );
```

SQL Clauses



GROUP BY

- SQL GROUP BY statement is used to arrange identical datainto groups.
- The GROUP BY statement is used with the SQL SELECT statement.
- The GROUP BY statement follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.
- The GROUP BY statement is used with aggregation function.

Syntax

```
SELECT column
FROM table_name
WHERE conditions
GROUP BY column
ORDER BY column
```

Example

```
SELECT COMPANY, COUNT(*)
FROM PRODUCT_MAST
GROUP BY COMPANY;
```

HAVING

- HAVING clause is used to specify a search condition for a group or an aggregate.
- Having is used in a GROUP BY clause. If you are not using GROUP BY clause then you can use HAVING function like a WHERE clause

Syntax

```
SELECT column1, column2 FRO
M table_name
WHERE conditions
GROUP BY column1, column2
HAVING conditions
ORDER BY column1, column2;
```

Example

```
SELECT COMPANY, COUNT(*)
FROM PRODUCT_MAST
GROUP BY COMPANY
HAVING COUNT(*)>2;
```

ORDER BY

- The ORDER BY clause sorts the result-set in ascending or descending order.
- It sorts the records in ascending order by default. DESC keyword is used to sort the records in descending order.

Syntax

```
SELECT column1, column2
FROM table_name
WHERE condition
ORDER BY column1, column2... AS
C|DESC;
```

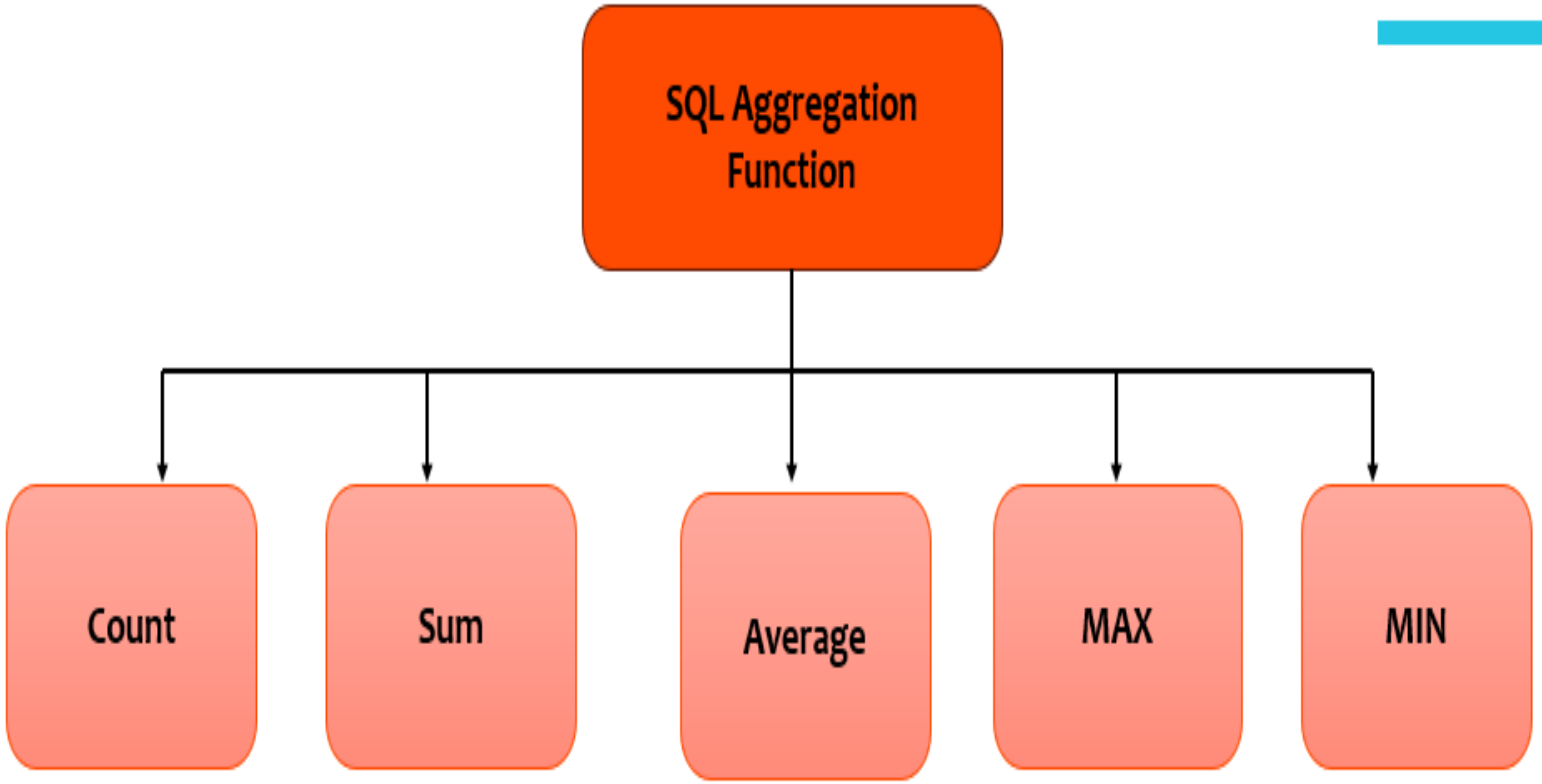
Example

```
SELECT *
FROM CUSTOMER
ORDER BY NAME;
```

OR

```
SELECT *
FROM        CUSTOMER
ORDER BY NAME DESC;
```

SQL Aggregate Functions



COUNT FUNCTION

- COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.
- COUNT function uses the COUNT(*) that returns the count of allthe rows in a specified table. COUNT(*) considers duplicate and Null.

Syntax

COUNT(*) or COUNT([ALL|DISTINCT] expression)

Example

- SELECT COUNT(*) FROM PRODUCT_MAST;
- SELECT COUNT(*) FROM PRODUCT_MAST; WHERE RATE>=20;
- SELECT COUNT(DISTINCT COMPANY) FROM PRODUCT_MAST;
- SELECT COMPANY, COUNT(*) FROM PRODUCT_MAST GROUP BY COMPANY;
- SELECT COMPANY, COUNT(*) FROM PRODUCT_MAST GROUP BY COMPANY HAVING COUNT(*)>2;

SUM FUNCTION

- Sum function is used to calculate the sum of allselected columns. It works on numeric fields only.

Syntax

SUM() or SUM([ALL|DISTINCT] expression)

Example

SELECT SUM(COST) FROM PRODUCT_MAST;

SUM() with WHERE

SELECT SUM(COST) FROM PRODUCT_MAST WHERE QTY>3;

SUM() with GROUP BY

```
SELECT SUM(COST)  FROM PRODUCT_MAST  WHERE QTY>3
GROUP BY COMPANY;
```

SUM() with HAVING

SELECT COMPANY, SUM(COST) FROM PRODUCT_MAST GROUP BY COM PANY HAVING SUM(COST)>=170;

AVG FUNCTION

- The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

Syntax

AVG() or AVG([ALL|DISTINCT] expression)

Example

SELECT AVG(COST) FROM PRODUCT_MAST;

MAX FUNCTION

- MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

Syntax

MAX() or MAX([ALL|DISTINCT] expression)

Example

SELECT MAX(RATE) FROM PRODUCT_MAST;

MIN FUNCTION

- MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column

Syntax

MIN() or MIN([ALL|DISTINCT] expression)

Example

SELECT MIN(RATE) FROM PRODUCT_MAST;

SQL JOIN

SQL, JOIN means "to combine two or more tables". In SQL, JOIN clause is used to combine the records from two or more tables in a database.

Types of SQL JOIN

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

INNER JOIN

In SQL, INNER JOIN selects records that have matching values in both tables as long as the condition is satisfied. It returns the combination of all rows from both the tables where the condition satisfies.

Syntax

```
SELECT table1.column1, table1.column2, table2.column1,....
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
```

Example

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
INNER JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

LEFT JOIN

The SQL left join returns all the values from left table and the matching values from the right table. If there is no matching join value, it will return NULL.

Syntax

```
SELECT table1.column1, table1.column2, table2.column1,....
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

Example

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
LEFT JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

RIGHT JOIN

In SQL, RIGHT JOIN returns all the values from the values from the rows of right table and the matched values from the left table. If there is no matching in both tables, it will return NULL.

Syntax

```
SELECT table1.column1, table1.column2, table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

Example

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
RIGHT JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

FULL JOIN

In SQL, FULL JOIN is the result of a combination of both left and right outer join. Join tables have all the records from both tables. It puts NULL on the place of matches not found.

Syntax

```
SELECT table1.column1, table1.column2, table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;
```

Example

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
FULL JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

SQL Set Operation

The SQL Set operation is used to combine the two or more SQL SELECT statements

Types of Set Operation

- Union
- Union All
- Intersect
- Minus

Union Operation

- The SQL Union operation is used to combine the result of two or more SQL SELECT queries.
- In the union operation, all the number of datatype and columns mustbe same in both the tables on which UNION operation is being applied.
- The union operation eliminates the duplicate rows from its resultset.

Syntax

```
SELECT column_name FROM table1
UNION
SELECT column_name FROM table2;
```

Example

```
SELECT * FROM First
UNION
SELECT * FROM Second;
```

Intersect Operation

- It is used to combine two SELECT statements. The Intersect operation returns the common rows from both the SELECT statements.
- In the Intersect operation, the number of datatype and columns must be the same.
- It has no duplicates and it arranges the data in ascending order by default.

Syntax

```
SELECT column_name FROM table1
INTERSECT
SELECT column_name FROM table2;
```

Example

```
SELECT * FROM First
INTERSECT
SELECT * FROM Second;
```

MINUS Operation

- It combines the result of two SELECT statements. Minus operator is used to display the rows which are present in the first query but absent in the second query.
- It has no duplicates and data arranged in ascending order by default.

Syntax

```
SELECT column_name FROM table1
MINUS
SELECT column_name FROM table2;
```

Example

```
SELECT * FROM First
MINUS
SELECT * FROM Second;
```