**JOINS:**

• SQLJoins are used to combined records of two or more tables together based on related columns.

• Joins are used to represent records of multiples tables into single table.

• By using JOINS, we can retrieve data from two or more tables based on logical relationship between the tables.

• SQL JOIN expand the result set horizontally.

**Types:**

1. **INNER JOIN/JOIN: •** It is used to retrieve common records from both the tables involved in join.

• It returns only matching rows for which the predicate evaluates to TRUE and rows for which predicate evaluate to FALSE or UNKNOWN are discarded.

Syntax: SELECT \* FROM TABLE 1 JOIN/ INNER JOIN TABLE 2 ON TABLE1.COLUMN\_NAME =

TABLE 2. COLUMN\_NAME WHERE CONDITION;

1. **LEFT OUTER JOIN/LEFT JOIN:** • It combine records from both the table and return all records from left hand side table and common records from right hand side table.

• The result set will contain NULL values if there are no matching rows in right hand

side table.

Syntax: SELECT \* FROM TABLE 1 LEFT JOIN TABLE 2 ON TABLE1.COLUMN\_NAME =

TABLE 2. COLUMN\_NAME WHERE CONDITION;

1. **RIGHT OUTER JOIN/RIGHT JOIN:** • It combine records from both the table and return all records from right hand side table and common records from left hand side table.

• The result set will contain NULL values if there are no matching rows in right hand side table.

Syntax: SELECT \* FROM TABLE 1 RIGHT JOIN TABLE 2 ON TABLE1.COLUMN\_NAME =

TABLE 2. COLUMN\_NAME WHERE CONDITION;

1. **FULL OUTER JOIN/FULL JOIN:** • It returns all records when there is a match in left or right table records.

• It selects records that have matching values in these columns and the remaining

rows from both tables.

Syntax: SELECT \* FROM TABLE 1 FULL JOIN TABLE 2 ON TABLE1.COLUMN\_NAME

TABLE 2. COLUMN\_NAME WHERE CONDITION;

1. **CROSS JOIN:** • It is a Cartesian product of two tables.

• It returns very large result set of database records.

• The result of CROSS JOIN will not make sense if we have a need of precise result

Set.

• It is not commonly used one join.

• It performs multiplication between the tables and returns each combination of

rows from both sides of tables.

• There is no need to define predicate with ON keyword.

Syntax: SELECT \* FROM TABLE 1 CROSS JOIN TABLE 2;

1. **SELF JOIN:** • A self-join is a join in which a table is join with itself.

**•** Self join can be viewed as a join of two copies of the same table.

• It is necessary that a table can have two that columns that have same values and

must be unique.

Syntax: SELECT t1. column1, t2. Column2, ............ FROM TABLE\_1 AS t1

INNER/OUTER JOIN TABLE\_1 AS t2 ON t1. coulmn1=t2.column2;

**SET Operators:**

• It is used to merge data from two or more tables of same kind.

• It operates on two results set of queries which comparing the complete rows between the results.

• SET operators expand the result set vertically.

**Types:**

1. **UNION:** • It is used to combine two or more table and return single result set.

• It removes all duplicate rows from a table.

• Every SELECT statement within UNION must have the same number of columns.

• The columns must also have similar data types.

• The columns in every SELECT statement must also be in the same order.

• It sorts data in ascending order by default.

• Return DISTINCT records.

• Perform Distinct operation.

• Slower than UNION ALL.

SYNTAX: SELECT column\_name(s) FROM table1  
 UNION  
 SELECT column\_name(s) FROM table2;

1. **UNION ALL:** • It is used to combine two or more table and return single result set.

• It doesn’t remove duplicate rows from a table.

• Return duplicate records.

• Doesn’t return distinct records.

• Faster than UNION.

SYNTAX: SELECT column\_name(s) FROMtable1  
 UNION ALL  
 SELECT column\_name(s) FROM table2;

1. **INTERSECT: •** It retrieve the common unique records or rows from the first and second query.

• It removes duplicate rows.

Syntax: SELECTcolumn\_name(s) FROM table1  
 INTERSECT  
 SELECT column\_name(s) FROM table2;

1. **EXCEPT:** • It returns the rows from the first query which are not found in second query.

• It returns unique and distinct rows from the first query.

Syntax: SELECTcolumn\_name(s) FROM table1  
 EXCEPT  
 SELECT column\_name(s) FROM table2;

**SUBQUERIES:**

• A subquery is a query within another query.

• It is also known as nested query or inner query.

• We can use subqueries in INSERT, SELECT, UPDATE and DELETE statements.

• JOINS are more efficient than subqueries.

• A subquery must enclose in parenthesis.

• A subquery must include a SELECT clause and a FROM clause.

• A subquery must include optional WHERE, GROUP BY, and HAVING clauses.

• ORDER BY can only be specified when TOP is also specified.

• A subquery can nest up to 32 levels.

• A subquery can return different form of results which can be single value, multiple value in a single column or even a multi-column table result.

• We can use comparison operator with subqueries and another relation operator (ALL, ANY, IN).

• Whenever we want to compare columns to a list of value rather than a single value, in that case we can use IN, ALL, ANY operators.

Syntax: SELECTcolumn1, column2, .......

FROM table\_name  
WHERE <expression> <operator>  
 (SELECT column\_name(s) FROM table\_name);

**Types:**

1. **Self-contained Subquery:** • Self-contained Subqueries are subqueries that have no dependency on outer query.
2. **Corelated Subquery:** • It is the one that depends on the outer query for conditionally filters rows based on, the inner query.