Summary

Contents

[WHERE 2](#_Toc187252204)

[GROUP BY 2](#_Toc187252205)

[HAVING 2](#_Toc187252206)

[ORDER BY 2](#_Toc187252207)

[DISTINCT 3](#_Toc187252208)

[LIMIT 3](#_Toc187252209)

[OFFSET 3](#_Toc187252210)

[JOIN 3](#_Toc187252211)

[UNION 4](#_Toc187252212)

[CASE 4](#_Toc187252213)

[ALIAS 5](#_Toc187252214)

[SUBQUERIES 5](#_Toc187252215)

[IN and NOT IN 5](#_Toc187252216)

[BETWEEN 6](#_Toc187252217)

[LIKE and ILIKE (Pattern Matching) 6](#_Toc187252218)

[EXISTS and NOT EXISTS 6](#_Toc187252219)

[WINDOW FUNCTIONS 7](#_Toc187252220)

[COALESCE 7](#_Toc187252221)

[NULL Handling 8](#_Toc187252222)

[Indexes 8](#_Toc187252223)

FROM

* FROM: Specifies the table from which to retrieve the data.
  + Explanation: The FROM clause identifies the table or tables from which the data will be selected.
    - Example: FROM STUDENTS

## WHERE

* WHERE: Specifies the rows to be considered based on a condition.
  + Explanation: The WHERE clause filters the rows based on a specified condition.
    - Example: WHERE AGE > 18

## GROUP BY

* GROUP BY: Combines rows that have the same values in specified columns into aggregated data.
  + Explanation: The GROUP BY clause groups rows that have the same values in specified columns, allowing aggregate functions (like COUNT, SUM, AVG) to be performed on these groups.
    - Example: GROUP BY DEPARTMENT

## HAVING

* HAVING: Filters groups based on a condition applied to aggregated data.
  + Explanation: The HAVING clause is used to filter groups created by the GROUP BY clause based on a condition applied to the aggregated data.
    - Example: HAVING COUNT(\*) > 5

## ORDER BY

* ORDER BY: Specifies the order in which the result set should be sorted.
  + Explanation: The ORDER BY clause sorts the result set based on one or more columns, either in ascending (ASC) or descending (DESC) order.
    - Example: ORDER BY NAME ASC

## DISTINCT

* SELECT DISTINCT: Removes duplicate rows from the result set.
  + Example: SELECT DISTINCT DEPARTMENT FROM EMPLOYEES;

## LIMIT

* LIMIT: Restricts the number of rows returned in the result set (often used with OFFSET).
  + Example: SELECT \* FROM EMPLOYEES LIMIT 10;

## OFFSET

* OFFSET: Specifies the number of rows to skip before starting to return rows.
  + Example: SELECT \* FROM EMPLOYEES LIMIT 10 OFFSET 5;

## JOIN

* JOIN: Combines rows from two or more tables based on a related column between them.
* Types:
  + INNER JOIN: Returns rows that have matching values in both tables.
    - Example: SELECT \* FROM EMPLOYEES INNER JOIN DEPARTMENTS ON EMPLOYEES.DEPARTMENT\_ID = DEPARTMENTS.ID;
  + LEFT JOIN (or LEFT OUTER JOIN): Returns all rows from the left table and matched rows from the right table.
    - Example: SELECT \* FROM EMPLOYEES LEFT JOIN DEPARTMENTS ON EMPLOYEES.DEPARTMENT\_ID = DEPARTMENTS.ID;
  + RIGHT JOIN (or RIGHT OUTER JOIN): Returns all rows from the right table and matched rows from the left table.
    - Example: SELECT \* FROM EMPLOYEES RIGHT JOIN DEPARTMENTS ON EMPLOYEES.DEPARTMENT\_ID = DEPARTMENTS.ID;
  + FULL JOIN (or FULL OUTER JOIN): Returns all rows when there is a match in either left or right table.
    - Example: SELECT \* FROM EMPLOYEES FULL JOIN DEPARTMENTS ON EMPLOYEES.DEPARTMENT\_ID = DEPARTMENTS.ID;

## UNION

* UNION: Combines the result sets of two or more SELECT statements (removes duplicates by default).
  + Example: SELECT NAME FROM EMPLOYEES UNION SELECT NAME FROM CONTRACTORS;

UNION ALL

* UNION ALL: Combines the result sets of two or more SELECT statements (includes duplicates).
  + Example: SELECT NAME FROM EMPLOYEES UNION ALL SELECT NAME FROM CONTRACTORS;

## CASE

* CASE: Adds conditional logic to the SELECT statement.
  + Example:

SELECT

    EMPLOYEE\_ID,

    SALARY,

    CASE

        WHEN SALARY > 50000 THEN 'HIGH'

        WHEN SALARY BETWEEN 30000

        AND 50000 THEN 'MEDIUM'

        ELSE 'LOW'

    END AS SALARY\_RANGE

FROM

    EMPLOYEES;

## ALIAS

* Description: Assigns a temporary name to a table or column.
* Example:

SELECT

    NAME AS EMPLOYEE\_NAME

FROM

    EMPLOYEES;

## SUBQUERIES

* Description: A query nested within another SQL query.
* Example:

SELECT

    NAME

FROM

    EMPLOYEES

WHERE

    DEPARTMENT\_ID = (

        SELECT ID

        FROM DEPARTMENTS

        WHERE NAME = 'HR'

    );

## IN and NOT IN

* Description: Checks if a value exists within a list or subquery.
* Example:

SELECT

    \*

FROM

    EMPLOYEES

WHERE

    DEPARTMENT\_ID

IN (1, 2, 3);

SELECT

    \*

FROM

    EMPLOYEES

WHERE

    DEPARTMENT\_ID

NOT IN (4, 5, 6);

## BETWEEN

* Description: Filters rows based on a range of values.
* Example:

SELECT

    \*

FROM

    EMPLOYEES

WHERE

    AGE BETWEEN 25

    AND 35;

## LIKE and ILIKE (Pattern Matching)

* LIKE: Performs pattern matching (case-sensitive).
* ILIKE: Performs pattern matching (case-insensitive, supported in PostgreSQL).
* Example:

SELECT

    \*

FROM

    EMPLOYEES

WHERE

    NAME LIKE 'A%'; -- Starts with 'A'

SELECT

    \*

FROM

    EMPLOYEES

WHERE

    NAME LIKE '%n'; -- Ends with 'n'

## EXISTS and NOT EXISTS

* Description: Checks for the existence (or non-existence) of rows returned by a subquery.
* Example:

SELECT

    NAME

FROM

    EMPLOYEES

WHERE

    EXISTS (

        SELECT

            1

        FROM

            DEPARTMENTS

        WHERE

            DEPARTMENTS.ID = EMPLOYEES.DEPARTMENT\_ID

    );

## WINDOW FUNCTIONS

* Description: Performs calculations across a set of table rows related to the current row.
* Example:

SELECT

    NAME,

    SALARY,

    RANK() OVER (

        ORDER BY

            SALARY DESC

    ) AS SALARY\_RANK

FROM

    EMPLOYEES;

## COALESCE

* Description: Returns the first non-null value from a list of expressions.
* Example:

SELECT

    NAME,

    COALESCE(MANAGER\_ID, 'No Manager') AS MANAGER

FROM

    EMPLOYEES;

## NULL Handling

* Description: SQL uses IS NULL and IS NOT NULL to handle null values.
* Example:

SELECT

    \*

FROM

    EMPLOYEES

WHERE

    MANAGER\_ID IS NULL;

SELECT

    \*

FROM

    EMPLOYEES

WHERE

    MANAGER\_ID IS NOT NULL;

## Indexes

### **Types of Indexes**

1. Single-Column Index:

* Created on a single column.
* Example:

CREATE INDEX

    IDX\_EMPLOYEE\_ID

ON

    EMPLOYEES(EMPLOYEE\_ID);

1. Composite Index:

* Created on multiple columns, used when queries filter on a combination of columns.
* Example:

CREATE INDEX

    IDX\_NAME\_DEPARTMENT

ON

    EMPLOYEES(NAME, DEPARTMENT\_ID);

1. Unique Index:

* Ensures all values in the indexed column(s) are unique.
* Example:

CREATE UNIQUE INDEX

    IDX\_UNIQUE\_EMAIL

ON

    EMPLOYEES(EMAIL);

1. Full-Text Index:

* Used for searching large text fields.
* Example:

CREATE FULLTEXT INDEX

    IDX\_FULLTEXT\_BIO

ON

    EMPLOYEES(BIO);