

PostgreSQL-Specific SQL Features

Functions, Statements, and
Commands Not Available in MySQL

Advanced Window Functions

- PostgreSQL supports FILTER clauses and DISTINCT in aggregate functions, e.g.,
- `SELECT SUM(value) FILTER (WHERE value > 10)
AS filtered_sum FROM table_name;`

Common Table Expressions (CTEs) with Recursive Queries

- Recursive CTEs allow hierarchical data processing, e.g.,
- `WITH RECURSIVE cte_name AS (`
- `SELECT id, parent_id FROM table_name`
 `WHERE parent_id IS NULL`
- `UNION ALL`
- `SELECT t.id, t.parent_id FROM table_name t`
- `INNER JOIN cte_name c ON t.parent_id =`
 `c.id`
- `)`

Full JSON/JSONB Support

- PostgreSQL provides comprehensive JSON/JSONB support with operators like `->`, `->>`, and `#>`,
- and functions like `jsonb_set` and `jsonb_array_elements`, e.g.,
- `SELECT data->'key' AS value FROM json_table;`

Table Inheritance

- Allows tables to inherit columns and constraints from parent tables, e.g.,
- `CREATE TABLE parent_table (id SERIAL PRIMARY KEY, name TEXT);`
- `CREATE TABLE child_table () INHERITS (parent_table);`

Custom Data Types

- PostgreSQL supports custom data types like ENUM and ranges, e.g.,
- `CREATE TYPE mood AS ENUM ('happy', 'sad', 'neutral');`

Range Data Types

- Supports range types like int4range and daterange, e.g.,
- `SELECT * FROM range_table WHERE range_column @> 5;`

Full-Text Search

- PostgreSQL offers advanced full-text search with `to_tsvector` and `to_tsquery`, e.g.,
- `SELECT * FROM articles WHERE
to_tsvector('english', content) @@
to_tsquery('search_term');`

Materialized Views

- Supports materialized views that can be refreshed manually, e.g.,
- `CREATE MATERIALIZED VIEW view_name AS
SELECT * FROM table_name;`
- `REFRESH MATERIALIZED VIEW view_name;`

Extensive Indexing Options

- PostgreSQL supports advanced indexing methods like GIN, SP-GiST, and partial indexes, e.g.,
- `CREATE INDEX idx_partial ON table_name (column_name) WHERE column_name IS NOT NULL;`

Stored Procedures with Transaction Control

- Allows transaction control within stored procedures, e.g.,
- `CREATE OR REPLACE PROCEDURE my_proc()`
- `LANGUAGE plpgsql AS $$`
- `BEGIN`
- `INSERT INTO table_name VALUES (1);`
- `COMMIT;`
- `INSERT INTO table_name VALUES (2);`
- `END; $$;`

Foreign Data Wrappers (FDW)

- Connects to external data sources using FDWs, e.g.,
- `CREATE EXTENSION postgres_fdw;`
- `CREATE SERVER foreign_server FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host 'localhost', dbname 'foreign_db');`

Advanced Locking Options

- Supports row-level locking with granularity, e.g.,
- `SELECT * FROM table_name FOR NO KEY UPDATE;`

EXCLUDE Constraints

- Prevents overlapping ranges or values with EXCLUDE constraints, e.g.,
- CREATE TABLE bookings (
 - room INT,
 - tsrange DATERANGE,
 - EXCLUDE USING gist (room WITH =, tsrange WITH &&)
-);

Array Data Types

- Supports arrays with functions to query and manipulate them, e.g.,
- `SELECT ARRAY[1, 2, 3] AS my_array;`

Event Triggers

- Runs code in response to DDL events like `CREATE TABLE`.