Admissions & Support:
support@mockattempt.com
Phone / WhatsApp: +91 98765 43210
Course Code: PGSQ • **Updated:** 01 Nov 2026
https://mockattempt.com/courses/postgresql-training

PGSQ • RELATIONAL DATABASE INTERNALS & SQL ENGINEERING | **BEGINNER TO PRINCIPAL POSTGRESQL DBA & ARCHITECT**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

PostgreSQL Training

Master PostgreSQL 17 from MVCC Snapshot Isolation, WAL Durability, `pg_hba.conf` & Multi-Tenant `search_path` Schemas to Exact `NUMERIC`, `BIGINT GENERATED ALWAYS AS IDENTITY`, `TIMESTAMPTZ`, `INTERVAL`, `UUID`, `JSONB` GIN Indexing, TOAST `TEXT` & Transactional DDL — with runnable SQL scripts, 10×50 MCQ Mock Tests per topic, and Spoken Voice + Gemini AI Interviews at ₹2,999.

Duration: 15–17 Weeks • Live + Hands-On Projects (206 Hrs) • **Modules:** 16 • **Live Topics:** 42 • **Labs & Tasks:** 84 • **Mocks / AI Vivas:** 244 / 244 • **Projects:** 16 Module + 1 Live Client Prod

Fixed Non-Negotiable Tuition Fee: Base Fee **₹2,999** + 18% GST (₹540) = **Total Incl. GST: ₹3,539**

Live Batch Tracks: Weekday Live Online Batch (Mon – Fri) (Monday to Friday (5 Live Interactive Classes / Week) • 1.5 Hours / Day (60 Mins Live Architecture & Coding + 30 Mins Hands-On Lab) • 17–19 Weeks Fast-Track) | Weekend Intensive Live Online Batch (Sat – Sun) (Saturday & Sunday (Designed for Working Professionals & College Students) • 3.0 Hours / Day (2 Hrs Deep-Dive Live Coding + 1 Hr Guided Project Build) • 19–21 Weeks Weekend Track)

Core Technologies & Engineering Stack: PostgreSQL 17 Engine & `psql` • MVCC, WAL & Autovacuum Internals • PgBouncer, `pg_hba.conf` & SCRAM-SHA-256 • Schemas, RBAC & `search_path` Hardening • `NUMERIC`, `BIGINT IDENTITY` & `TIMESTAMPTZ` • `UUID` & Binary `JSONB` + GIN Indexes • `TEXT` TOAST Storage & Transactional DDL

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Understand PostgreSQL's Multi-Version Concurrency Control (MVCC), Write-Ahead Logging (WAL), TOAST storage, and why readers never block writers
2. Harden PostgreSQL clusters with `scram-sha-256` in `pg_hba.conf`, PgBouncer pooling, PostgreSQL 15+ `public` schema lockdown, and explicit `search_path` rules
3. Choose optimal PostgreSQL data types (`NUMERIC` vs float, `TEXT` vs `CHAR(n)`, `BIGINT GENERATED ALWAYS AS IDENTITY` vs legacy `SERIAL`, `TIMESTAMPTZ`, `UUID`, and GIN-indexed `JSONB`)
4. Clear 200 Timed Mock Exams (10,000 PostgreSQL MCQs) and 200 Spoken Voice + Gemini AI PostgreSQL DBA & Backend Interviews
5. Rich PostgreSQL data modeling: strict constraints, JSONB hybrid documents, arrays, UUIDv7, and Row-Level Security (RLS).
6. Precision indexing: B-Tree, GIN (JSONB & Full-Text Search), GiST, BRIN, partial/expression indexes, and `pgvector` HNSW.
7. MVCC concurrency internals, WAL durability, autovacuum bloat prevention, declarative table partitioning, and CTE windowing.
8. Deep query plan optimization using `EXPLAIN (ANALYZE, BUFFERS)` and `pg_stat_statements` bottleneck analysis.

Eligibility & Prerequisites: Basic familiarity with tables, rows, and introductory SQL queries — covers PostgreSQL 17 architecture and SQL standards step-by-step from scratch • Any laptop/PC running PostgreSQL 16/17 (local installer or Docker `postgres:17`), `psql`, and DBeaver / pgAdmin 4 / DataGrip • No prior PostgreSQL industry experience required — Module 1 starts from environment setup, core mental models, and plain-English fundamentals. • Basic computer & internet familiarity on a Windows, macOS, or Linux laptop (8 GB+ RAM recommended for smooth live coding labs). • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews.

2. DETAILED MODULE & TOPIC-WISE SYLLABUS (16 MODULES • 42 LIVE TOPICS)

MODULE 1 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: PostgreSQL Object-Relational Architecture, MVCC Internals & Enterprise Advantages

Explore PostgreSQL's process/memory architecture (`shared_buffers`, WAL, 8 KB heap pages), MVCC non-blocking snapshot isolation, Transactional DDL, and rich extensibility (`pgvector`, PostGIS).

Objectives: ✓ Explain how Multi-Version Concurrency Control (MVCC) `xmin`/`xmax` tuple headers ensure readers never block writers and writers never block readers • ✓ Trace Write-Ahead Logging (WAL) crash recovery and why Autovacuum must reclaim dead tuples and prevent XID wraparound • ✓ Leverage

PostgreSQL's standout advantages: Transactional DDL, `RETURNING`, CTEs, Partial/Expression/GIN indexes, and extensions

1.1 PostgreSQL 17 Architecture — Object-Relational Engine, 8 KB Pages, WAL & MVCC Snapshot Isolation [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL 17 Architecture — Object-Relational Engine, 8 KB Pages, WAL & MVCC Snapshot Isolation in real PostgreSQL systems — specifically: How Po...

1.2 Advantages of PostgreSQL — Transactional DDL, Partial/GIN Indexes, Rich Types & Extensions [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Advantages of PostgreSQL — Transactional DDL, Partial/GIN Indexes, Rich Types & Extensions in real PostgreSQL systems — specifically: Why PostgreS...

Module 1 Hands-On Capstone Project (100 Marks): MVCC Tuple Visibility & Atomic `INSERT ... ON CONFLICT ... RETURNING` Ledger — Write a PostgreSQL 17 SQL script inspecting live vs dead tuple counters (`pg_stat_user_tables`), transaction IDs (`pg_current_xact_id()`), and atomic idempotent upserts via `ON CONFLICT DO U...

MODULE 2 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: Cluster Initialization (`initdb`), `postgresql.conf` Tuning & Secure Client Connectivity

Initialize `$PGDATA` clusters cleanly, tune `shared_buffers`/`work_mem`, lock down `pg_hba.conf` with `scram-sha-256` TLS, and scale connections via PgBouncer.

Objectives: ✓ Configure `$PGDATA`, `initdb --data-checksums`, and memory settings (`shared_buffers`, `effective_cache_size`, `work_mem`) in `postgresql.conf` • ✓ Enforce `hostssl` + `scram-sha-256` authentication inside `pg_hba.conf` (eliminating `trust` and legacy `md5`) • ✓ Prevent per-connection backend process exhaustion (~5–10 MB RAM per forked Postgres backend) using PgBouncer transaction pooling

2.1 Installing & Tuning PostgreSQL 17 — `initdb --data-checksums`, `$PGDATA` & `postgresql.conf` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Installing & Tuning PostgreSQL 17 — `initdb --data-checksums`, `$PGDATA` & `postgresql.conf` in real PostgreSQL systems — specifically: Provisioni...

2.2 Connecting to PostgreSQL — Port 5432, `pg_hba.conf` (`scram-sha-256`), TLS & PgBouncer Pooling [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Connecting to PostgreSQL — Port 5432, `pg_hba.conf` (`scram-sha-256`), TLS & PgBouncer Pooling in real PostgreSQL systems — specifically: Securing...

Module 2 Hands-On Capstone Project (100 Marks): Cluster Memory Parameter Auditor & Active Session Diagnostics (`pg_stat_activity`) — Write a PostgreSQL DBA diagnostic script auditing key runtime settings (`pg_settings`), idle-in-transaction timeouts, and active client backend states (`pg_stat_activity`).

MODULE 3 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: PostgreSQL Lexical Syntax, Identifier Folding & `CREATE DATABASE` Internals

Master ANSI string literals vs double-quoted case-sensitive identifiers, dollar-quoting (`$$`), logical `SELECT` evaluation order, and template-based `CREATE DATABASE` isolation.

Objectives: ✓ Avoid mixed-case double-quoted identifier traps (`"UserId"` vs unquoted lowercase folding `user_id`) • ✓ Use single quotes `'...'` for string values, `$1, $2` for parameterized SQL, and `$$...$$` for clean function/procedural blocks • ✓ Understand `template1` vs `template0` cloning, `UTF8` ICU/C collations, and cross-database isolation during `CREATE DATABASE`

3.1 PostgreSQL SQL Syntax — Unquoted Lowercase Folding, Single vs Double Quotes, `$$` & Clause Order [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL SQL Syntax — Unquoted Lowercase Folding, Single vs Double Quotes, `$$` & Clause Order in real PostgreSQL systems — specifically: Master...

3.2 PostgreSQL `CREATE DATABASE` — `template0` vs `template1`, `UTF8` ICU Collations & Cross-DB Isolation [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `CREATE DATABASE` — `template0` vs `template1`, `UTF8` ICU Collations & Cross-DB Isolation in real PostgreSQL systems — specifically: C...

Module 3 Hands-On Capstone Project (100 Marks): Snake-Case Identifier Standardizer & Database Collation / Limit Hardener — Write a clean snake_case PostgreSQL SQL query using CTEs (`WITH`), dollar-quoted strings, and `COALESCE` alongside role/database `CONNECTION LIMIT` and `STATEMENT_TIMEOUT` governance.

MODULE 4 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: Namespaces (`Schemas`), `search_path` Security & PostgreSQL 15+ `public` Lockdown

Organize multi-domain or multi-tenant databases cleanly using PostgreSQL Schemas, understand `search_path` object resolution, and enforce `CREATE SCHEMA ... AUTHORIZATION` least-privilege rules.

Objectives: ✓ Contrast PostgreSQL Schemas (logical namespaces inside ONE database supporting cross-schema joins & foreign keys) with MySQL where `SCHEMA` is a synonym for `DATABASE` • ✓ Understand PostgreSQL 15+ security hardening (revoking default `CREATE ON SCHEMA public FROM PUBLIC` to prevent CVE-2018-1058 `search_path` trojan hijacks) • ✓ Provision isolated domain schemas (`billing`, `catalog`, `audit`) via `CREATE SCHEMA IF NOT EXISTS ... AUTHORIZATION` and `GRANT USAGE`

4.1 PostgreSQL Schemas & `search_path` — Namespace Resolution, PG 15+ `public` Hardening & Multi-Tenancy [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL Schemas & `search_path` — Namespace Resolution, PG 15+ `public` Hardening & Multi-Tenancy in real PostgreSQL systems — specifically: Ho...

4.2 PostgreSQL `CREATE SCHEMA` — `AUTHORIZATION`, Inline Atomic DDL & Default Privileges [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `CREATE SCHEMA` — `AUTHORIZATION`, Inline Atomic DDL & Default Privileges in real PostgreSQL systems — specifically: Provisioning role-...

Module 4 Hands-On Capstone Project (100 Marks): Multi-Domain Schema Isolation (`billing` + `catalog`) with Explicit Cross-Schema Foreign Keys — Create isolated `catalog` and `billing` schemas within a single database, link `billing.invoices` to `catalog.courses` via schema-qualified Foreign Keys, and lock down `search_path`.

MODULE 5 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: Type System Architecture & Exact Financial Arithmetic (`NUMERIC`)

Master PostgreSQL's rich native type catalog (column alignment padding, domains, arrays, ranges) and prevent IEEE-754 floating-point rounding bugs in financial ledgers using exact `NUMERIC(p, s)`.

Objectives: ✓ Select idiomatic PostgreSQL native types (`inet`, `cidr`, `tstzrange`, `bigint`, `timestamptz`, `uuid`, `jsonb`) over stringly-typed `VARCHAR(255)` columns • ✓ Understand `pg_column_size()` and CPU word-alignment ("column tetris": placing 8-byte `BIGINT`/`TIMESTAMPTZ` columns before 4-byte `INT` and 1-byte `BOOLEAN` columns) • ✓ Guarantee zero penny rounding drift in wallets/invoices using base-10000 exact `NUMERIC(precision, scale)` instead of `REAL`/`DOUBLE PRECISION` (`0.1 + 0.2 != 0.3`)

5.1 PostgreSQL Data Types Overview — Rich Native Types (`inet`, Ranges, Arrays), Casting (`::`) & Column Alignment [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL Data Types Overview — Rich Native Types (`inet`, Ranges, Arrays), Casting (`::`) & Column Alignment in real PostgreSQL systems — specif...

5.2 PostgreSQL `NUMERIC` / `DECIMAL` — Exact Base-10 Arithmetic vs IEEE-754 Float Drift, Scale & `NaN` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `NUMERIC` / `DECIMAL` — Exact Base-10 Arithmetic vs IEEE-754 Float Drift, Scale & `NaN` in real PostgreSQL systems — specifically: Why...

Module 5 Hands-On Capstone Project (100 Marks): IEEE-754 Float Drift Proof & Exact `NUMERIC(14, 2)` Double-Entry Wallet Ledger — Demonstrate binary `FLOAT8` (`0.1 + 0.2`) inaccuracy versus exact `NUMERIC` arithmetic and build a `CHECK (balance >= 0)` wallet ledger calculating inclusive/exclusive GST taxes.

MODULE 6 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Character Storage Internals (`VARCHAR` vs `CHAR` vs `TEXT`) & Three-Valued `BOOLEAN` Logic

Understand why PostgreSQL `VARCHAR(n)` and `TEXT` share identical `varlena` performance (while `CHAR(n)` wastes space!), and avoid catastrophic `NOT IN (NULL)` bugs with 3-Valued Boolean Logic (`IS DISTINCT FROM`).

Objectives: ✓ Understand PostgreSQL `varlena` internals: why `CHAR(n)` is padded slow legacy debt, while `TEXT` + `CHECK (char_length(col) <= n)` allows zero-lock length changes • ✓ Master SQL Three-Valued Logic (`TRUE`, `FALSE`, `NULL`/Unknown) and why `NULL = NULL` and `NULL <> true` both evaluate to `NULL` • ✓ Write null-safe predicates using `IS TRUE`, `IS NOT TRUE`, `IS DISTINCT FROM`, and `NOT EXISTS` (never `NOT IN` over nullable columns!)

6.1 PostgreSQL `VARCHAR(n)` vs `CHAR(n)` vs `TEXT` — `varlena` Internals, Character Count & Zero-Lock Limits [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `VARCHAR(n)` vs `CHAR(n)` vs `TEXT` — `varlena` Internals, Character Count & Zero-Lock Limits in real PostgreSQL systems — specifically...

6.2 PostgreSQL `BOOLEAN` — 1-Byte Storage, Three-Valued Logic (`TRUE`/`FALSE`/`NULL`) & `IS DISTINCT FROM` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `BOOLEAN` — 1-Byte Storage, Three-Valued Logic (`TRUE`/`FALSE`/`NULL`) & `IS DISTINCT FROM` in real PostgreSQL systems — specifically:...

Module 6 Hands-On Capstone Project (100 Marks): Zero-Lock `TEXT` Constraint Validator & Null-Safe Boolean Filter Engine — Demonstrate PostgreSQL's 3-valued boolean trap where `WHERE is_banned <> true` silently drops `NULL` rows (fixed via `IS NOT TRUE`) and `NOT IN (SELECT nullable_col ...)` returns 0 rows (fix...

MODULE 7 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: 64-Bit Primary Keys (`BIGINT`) & SQL-Standard `GENERATED AS IDENTITY` vs Legacy `SERIAL`

Prevent 2.14-billion `INT4` primary key exhaustion outages by defaulting to `BIGINT` (`int8`), and replace legacy `SERIAL` macros with SQL-standard `GENERATED ALWAYS AS IDENTITY`.

Objectives: ✓ Understand why greenfield primary keys and high-volume foreign keys should use 8-byte `BIGINT` (avoiding multi-hour table-rewrite `INT` overflow incidents at 2,147,483,647) • ✓ Replace legacy PostgreSQL `SERIAL` / `BIGSERIAL` pseudo-types with SQL:2003 `BIGINT GENERATED ALWAYS AS IDENTITY` • ✓ Recognize why sequence-backed IDs intentionally exhibit gaps on rolled-back transactions (non-transactional sequence caching)

7.1 PostgreSQL `BIGINT` (`int8`) — 64-Bit Range, Preventing `2.14B` `INT4` Exhaustion & FK Type Matching [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `BIGINT` (`int8`) — 64-Bit Range, Preventing `2.14B` `INT4` Exhaustion & FK Type Matching in real PostgreSQL systems — specifically: Wh...

7.2 PostgreSQL `SERIAL` vs `GENERATED ALWAYS AS IDENTITY` — Sequence Mechanics, Gaps & `setval` Desync [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `SERIAL` vs `GENERATED ALWAYS AS IDENTITY` — Sequence Mechanics, Gaps & `setval` Desync in real PostgreSQL systems — specifically: Unde...

Module 7 Hands-On Capstone Project (100 Marks): Overflow-Proof `BIGINT GENERATED ALWAYS AS IDENTITY` Order Sequence & Gap Behaviour Lab — Build an order ledger using `BIGINT GENERATED ALWAYS AS IDENTITY` that rejects accidental manual ID overrides (unless `OVERRIDING SYSTEM VALUE` is explicit) and demonstrates why rolled-back...

MODULE 8 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: Timezone-Safe Temporal Engineering (`TIMESTAMPTZ` & `INTERVAL`)

Store instants unambiguously in UTC microseconds with `TIMESTAMPTZ` (never naive `TIMESTAMP`!), preserve B-Tree index SARGability on date ranges, and compute calendar-aware `INTERVAL` math.

Objectives: ✓ Understand that BOTH `TIMESTAMP` and `TIMESTAMPTZ` occupy 8 bytes, and `TIMESTAMPTZ` converts inputs to UTC microseconds since `2000-01-01` internally • ✓ Preserve B-Tree index scans using SARGable half-open range predicates (`created_at >= start AND created_at < end`) instead of `DATE(created_at) = ...` • ✓ Master 16-byte PostgreSQL `INTERVAL` arithmetic (separate months, days, and microseconds components), `date_trunc()`, and `AT TIME ZONE`

8.1 PostgreSQL `TIMESTAMPTZ` vs `TIMESTAMP` — 8-Byte UTC Microseconds, `AT TIME ZONE` & SARGable Ranges [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `TIMESTAMPTZ` vs `TIMESTAMP` — 8-Byte UTC Microseconds, `AT TIME ZONE` & SARGable Ranges in real PostgreSQL systems — specifically: Why...

8.2 PostgreSQL `INTERVAL` — 16-Byte Calendar/Day/Microsecond Triplet, DST Awareness & `justify_interval` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `INTERVAL` — 16-Byte Calendar/Day/Microsecond Triplet, DST Awareness & `justify_interval` in real PostgreSQL systems — specifically: Pe...

Module 8 Hands-On Capstone Project (100 Marks): DST-Safe Cohort Subscription Scheduler (`TIMESTAMPTZ` + `INTERVAL` + SARGable Ranges) — Write a timezone-safe PostgreSQL billing query computing `Asia/Kolkata` and `America/New_York` views over UTC `TIMESTAMPTZ` instants using SARGable range bounds and calendar `INTERVAL` renew...

MODULE 9 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: Native 16-Byte `UUID` Keys (`UUIDv4` vs `UUIDv7`) & Binary `JSONB` + GIN Indexing

Store compact 128-bit `UUID` identifiers natively (understanding random `UUIDv4` B-Tree page-split bloat vs time-ordered `UUIDv7`!), and accelerate semi-structured `JSONB` queries by 1000x with GIN `@>` indexes.

Objectives: ✓ Store UUIDs in PostgreSQL's 16-byte binary `UUID` type (never 36-byte `VARCHAR(36)`) and contrast random `gen_random_uuid()` (UUIDv4) vs sequential time-ordered `UUIDv7` (`uuidv7()` in PG 18 / app-generated) • ✓ Always choose parsed binary `JSONB` over raw text-reparsing `JSON` unless exact whitespace/duplicate-key preservation is required for raw webhook archiving • ✓ Index `JSONB` columns with `USING gin (payload jsonb_path_ops)` and query via containment `@>` (`?`, `?|`, `?&`) instead of non-indexed `->>` full-table scans

9.1 PostgreSQL `UUID` — 16-Byte Binary Storage, `gen_random_uuid()`, & `UUIDv4` B-Tree Fragmentation vs `UUIDv7` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `UUID` — 16-Byte Binary Storage, `gen_random_uuid()`, & `UUIDv4` B-Tree Fragmentation vs `UUIDv7` in real PostgreSQL systems — specific...

9.2 PostgreSQL `JSON` vs `JSONB` — Decomposed Binary Storage, `->` vs `->>`, Containment `@>` & GIN Indexes [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `JSON` vs `JSONB` — Decomposed Binary Storage, `->` vs `->>`, Containment `@>` & GIN Indexes in real PostgreSQL systems — specifically:...

Module 9 Hands-On Capstone Project (100 Marks): Native `UUID` Event Store with GIN-Indexed `JSONB` Containment (`@>`) Search — Create an event store keyed by native 16-byte `UUID` (`DEFAULT gen_random_uuid()`) with a `JSONB` metadata payload indexed via `GIN (meta jsonb_path_ops)` and queried cleanly using `@>` and...

MODULE 10 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: Large `TEXT` TOAST Compression / Full-Text Search & Zero-Accident `ALTER SCHEMA` Governance

Understand how PostgreSQL stores values up to 1 GB inside 8 KB pages via TOAST (`pglz` / `lz4`), accelerate search with `tsvector` + `pg_trgm` GIN indexes, and govern `ALTER SCHEMA` safely without breaking `SECURITY DEFINER` / view dependencies.

Objectives: ✓ Understand PostgreSQL TOAST (The Oversized-Attribute Storage Technique): inline < ~2 KB, `lz4`/`pglz` compression, and out-of-line chunking up to 1 GB (plus why `SELECT *` hurts TOASTed tables!) • ✓ Accelerate substring `ILIKE '%term%'` and full-text search using `pg_trgm` (`gin_trgm_ops`) and `tsvector` @@ `tsquery` GIN indexes instead of sequential scans • ✓ Execute `ALTER SCHEMA ... RENAME TO` and `OWNER TO` safely while auditing OID-resolved views vs dynamic-SQL / `search_path` references

10.1 PostgreSQL `TEXT` — TOAST Internals (`lz4` Compression, Out-of-Line Chunks), `SELECT *` Cost & Full-Text Search [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `TEXT` — TOAST Internals (`lz4` Compression, Out-of-Line Chunks), `SELECT *` Cost & Full-Text Search in real PostgreSQL systems — speci...

10.2 PostgreSQL `ALTER SCHEMA` — `RENAME TO`, `OWNER TO`, OID Dependency Tracking & Zero-Downtime Cutovers [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply PostgreSQL `ALTER SCHEMA` — `RENAME TO`, `OWNER TO`, OID Dependency Tracking & Zero-Downtime Cutovers in real PostgreSQL systems — specifically: M...

Module 10 Hands-On Capstone Project (100 Marks): Full-Text `tsvector` Search Engine & Schema Ownership / Rename Migration Script — Build a PostgreSQL 17 documentation search table using a stored generated `tsvector` column (`GENERATED ALWAYS AS (to_tsvector('english', ...)) STORED`) and execute an atomic transactional `...

MODULE 11 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

MVCC Heap Internals, VACUUM/Autovacuum Tuning & Lock Contention

Production-grade PostgreSQL Training engineering module covering MVCC Heap Internals, VACUUM/Autovacuum Tuning & Lock Contention with hands-on labs and capstone build.

Objectives: ✓ Tuple Visibility (xmin/xmax), Dead Tuples, HOT Updates & Table Bloat — Purpose: Design update patterns and fillfactor settings that keep PostgreSQL heaps lean and fast. • ✓ Tuning Per-Table Autovacuum & Preventing Transaction ID Wraparound — Purpose: Ensure high-churn queue/session tables vacuum continuously without emergency downtime. • ✓ Lock Modes, lock_timeout Guards & Zero-Downtime DDL Habits — Purpose: Add columns, NOT NULL check constraints, and CREATE INDEX CONCURRENTLY without locking traffic!. • ✓ SKIP LOCKED Transactional Queues & Advisory Locks — Purpose: Build rock-solid concurrent job claimers right inside PostgreSQL.

11.1 Tuple Visibility (xmin/xmax), Dead Tuples, HOT Updates & Table Bloat [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Tuple Visibility (xmin/xmax), Dead Tuples, HOT Updates & Table Bloat in real PostgreSQL systems — specifically: Design update patterns and fillfac...

11.2 Tuning Per-Table Autovacuum & Preventing Transaction ID Wraparound [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Tuning Per-Table Autovacuum & Preventing Transaction ID Wraparound in real PostgreSQL systems — specifically: Ensure high-churn queue/session tabl...

11.3 Lock Modes, lock_timeout Guards & Zero-Downtime DDL Habits [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Lock Modes, lock_timeout Guards & Zero-Downtime DDL Habits in real PostgreSQL systems — specifically: Add columns, NOT NULL check constraints, and...

11.4 SKIP LOCKED Transactional Queues & Advisory Locks [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply SKIP LOCKED Transactional Queues & Advisory Locks in real PostgreSQL systems — specifically: Build rock-solid concurrent job claimers right inside...

Module 11 Hands-On Capstone Project (100 Marks): Zero-Bloat High-Write Financial Ledger on PostgreSQL 16 — Design, implement, test, and document 'Zero-Bloat High-Write Financial Ledger on PostgreSQL 16' applying every production concept from Module 11 (MVCC Heap Internals, VACUUM/Autovacuum Tunin...

MODULE 12 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Advanced Indexing (B-Tree, GIN, GiST, BRIN) & EXPLAIN BUFFERS Mastery

Production-grade PostgreSQL Training engineering module covering Advanced Indexing (B-Tree, GIN, GiST, BRIN) & EXPLAIN BUFFERS Mastery with hands-on labs and capstone build.

Objectives: ✓ Reading EXPLAIN (ANALYZE, BUFFERS): Seq/Bitmap/Index-Only Scans & Sort Spills — Purpose: Spot disk work_mem spills and stale statistics immediately from execution plans. • ✓ Partial Indexes, Expression Indexes & Covering INCLUDE Columns — Purpose: Index only the 2% active unpaid orders for tiny RAM-resident sub-millisecond lookups. • ✓ JSONB Containment (@>), jsonb_path_ops GIN Indexes & Full-Text tsvector — Purpose: Query flexible attributes and ranked full-text search without adding separate search infra. • ✓ BRIN Indexes for Billion-Row Append-Only Logs & pg_stat_statements Audits — Purpose: Rank the top 10 DB CPU-consuming queries in production and fix them systematically.

12.1 Reading EXPLAIN (ANALYZE, BUFFERS): Seq/Bitmap/Index-Only Scans & Sort Spills [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Reading EXPLAIN (ANALYZE, BUFFERS): Seq/Bitmap/Index-Only Scans & Sort Spills in real PostgreSQL systems — specifically: Spot disk work_mem spills...

12.2 Partial Indexes, Expression Indexes & Covering INCLUDE Columns [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Partial Indexes, Expression Indexes & Covering INCLUDE Columns in real PostgreSQL systems — specifically: Index only the 2% active unpaid orders f...

12.3 JSONB Containment (@>), jsonb_path_ops GIN Indexes & Full-Text tsvector [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply JSONB Containment (@>), jsonb_path_ops GIN Indexes & Full-Text tsvector in real PostgreSQL systems — specifically: Query flexible attributes and r...

12.4 BRIN Indexes for Billion-Row Append-Only Logs & pg_stat_statements Audits [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply BRIN Indexes for Billion-Row Append-Only Logs & pg_stat_statements Audits in real PostgreSQL systems — specifically: Rank the top 10 DB CPU-consum...

Module 12 Hands-On Capstone Project (100 Marks): 100x Query Speedup Lab on 10M-Row JSONB + Relational Data — Design, implement, test, and document '100x Query Speedup Lab on 10M-Row JSONB + Relational Data' applying every production concept from Module 12 (Advanced Indexing (B-Tree, GIN, GiST, BRIN...

MODULE 13 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Declarative Partitioning, Row-Level Security (RLS) & pgvector AI Search

Production-grade PostgreSQL Training engineering module covering Declarative Partitioning, Row-Level Security (RLS) & pgvector AI Search with hands-on labs and capstone build.

Objectives: ✓ Declarative Range/Hash Partitioning, Partition Pruning & Instant Detach — Purpose: Manage 500GB+ time-series tables cleanly and drop old monthly partitions in 5ms. • ✓ Multi-Tenant Row-Level Security (RLS) Policies with Session Context — Purpose: Enforce tenant isolation inside the PostgreSQL kernel so app bugs never leak cross-tenant rows!. • ✓ Lateral Joins, Grouping Sets, Window Frames & PL/pgSQL Audit Triggers — Purpose: Compute rich analytical rollups and tamper-evident compliance audit trails. • ✓ pgvector Embeddings, HNSW Indexes & Hybrid Relational + Semantic Queries — Purpose: Power modern AI RAG retrieval directly inside ACID PostgreSQL transactions.

13.1 Declarative Range/Hash Partitioning, Partition Pruning & Instant Detach [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Declarative Range/Hash Partitioning, Partition Pruning & Instant Detach in real PostgreSQL systems — specifically: Manage 500GB+ time-series table...

13.2 Multi-Tenant Row-Level Security (RLS) Policies with Session Context [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Multi-Tenant Row-Level Security (RLS) Policies with Session Context in real PostgreSQL systems — specifically: Enforce tenant isolation inside the...

13.3 Lateral Joins, Grouping Sets, Window Frames & PL/pgSQL Audit Triggers [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Lateral Joins, Grouping Sets, Window Frames & PL/pgSQL Audit Triggers in real PostgreSQL systems — specifically: Compute rich analytical rollups a...

13.4 pgvector Embeddings, HNSW Indexes & Hybrid Relational + Semantic Queries [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply pgvector Embeddings, HNSW Indexes & Hybrid Relational + Semantic Queries in real PostgreSQL systems — specifically: Power modern AI RAG retrieval...

Module 13 Hands-On Capstone Project (100 Marks): Multi-Tenant RLS-Protected SaaS & Vector Search Database — Design, implement, test, and document 'Multi-Tenant RLS-Protected SaaS & Vector Search Database' applying every production concept from Module 13 (Declarative Partitioning, Row-Level Securit...

MODULE 14 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Streaming Replication, PgBouncer Pooling, WAL PITR & High Availability

Production-grade PostgreSQL Training engineering module covering Streaming Replication, PgBouncer Pooling, WAL PITR & High Availability with hands-on labs and capstone build.

Objectives: ✓ Transaction-Mode Connection Pooling with PgBouncer — Purpose: Multiplex 2,000 microservice pods smoothly onto 80 physical PostgreSQL backend processes. • ✓ Physical Streaming Replication, Replication Slots & Logical CDC — Purpose: Scale read replicas safely while monitoring slot lag to prevent WAL disk overflow. • ✓ Continuous WAL Archiving (pgBackRest) & Point-in-Time Recovery (PITR) — Purpose: Practice restoring a corrupted cluster to 14:32:10 UTC with verified integrity. • ✓ SCRAM-SHA-256, SSL Enforcement, Role Hierarchies & Patroni HA Failover — Purpose: Automate sub-15-second primary failover and pass enterprise DB security audits.

14.1 Transaction-Mode Connection Pooling with PgBouncer [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Transaction-Mode Connection Pooling with PgBouncer in real PostgreSQL systems — specifically: Multiplex 2,000 microservice pods smoothly onto 80 p...

14.2 Physical Streaming Replication, Replication Slots & Logical CDC [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Physical Streaming Replication, Replication Slots & Logical CDC in real PostgreSQL systems — specifically: Scale read replicas safely while monito...

14.3 Continuous WAL Archiving (pgBackRest) & Point-in-Time Recovery (PITR) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Continuous WAL Archiving (pgBackRest) & Point-in-Time Recovery (PITR) in real PostgreSQL systems — specifically: Practice restoring a corrupted cl...

14.4 SCRAM-SHA-256, SSL Enforcement, Role Hierarchies & Patroni HA Failover [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply SCRAM-SHA-256, SSL Enforcement, Role Hierarchies & Patroni HA Failover in real PostgreSQL systems — specifically: Automate sub-15-second primary f...

Module 14 Hands-On Capstone Project (100 Marks): HA Primary–Standby Cluster with PgBouncer & PITR Drills — Design, implement, test, and document 'HA Primary–Standby Cluster with PgBouncer & PITR Drills' applying every production concept from Module 14 (Streaming Replication, PgBouncer Pooling, WA...

MODULE 15 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

PostgreSQL MVCC Internals, Autovacuum Tuning, Row-Level Security & Partitioning

Self-explanatory list-and-bullet production mastery module covering PostgreSQL MVCC Internals, Autovacuum Tuning, Row-Level Security & Partitioning in PostgreSQL Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ MVCC Tuple Visibility, Dead Tuples, Autovacuum Tuning & Transaction ID Wraparound Safety — Purpose: Keep high-write PostgreSQL tables free from table/index bloat and vacuum stalls. • ✓ PostgreSQL Row-Level Security (RLS) Policies & Zero-Downtime CREATE INDEX CONCURRENTLY — Purpose: Enforce tenant isolation directly inside Postgres and evolve live schemas without write locks. • ✓ Declarative Range/Hash Partitioning, pg_stat_statements & Connection Pooling (PgBouncer) — Purpose: Scale to thousands of concurrent app connections and isolate slow queries scientifically. • ✓ Build, verify, and document 'RLS Multi-Tenant SaaS Schema with Declarative Partitioning & Bloat Monitor' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + Google Gemini AI Technical Interviews.

15.1 MVCC Tuple Visibility, Dead Tuples, Autovacuum Tuning & Transaction ID Wraparound Safety [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply MVCC Tuple Visibility, Dead Tuples, Autovacuum Tuning & Transaction ID Wraparound Safety in real PostgreSQL systems — specifically: Keep high-writ...

15.2 PostgreSQL Row-Level Security (RLS) Policies & Zero-Downtime CREATE INDEX CONCURRENTLY [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply PostgreSQL Row-Level Security (RLS) Policies & Zero-Downtime CREATE INDEX CONCURRENTLY in real PostgreSQL systems — specifically: Enforce tenant i...

15.3 Declarative Range/Hash Partitioning, pg_stat_statements & Connection Pooling (PgBouncer) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Declarative Range/Hash Partitioning, pg_stat_statements & Connection Pooling (PgBouncer) in real PostgreSQL systems — specifically: Scale to thous...

Module 15 Hands-On Capstone Project (100 Marks): RLS Multi-Tenant SaaS Schema with Declarative Partitioning & Bloat Monitor — Architect, implement, test, and document 'RLS Multi-Tenant SaaS Schema with Declarative Partitioning & Bloat Monitor' applying every bullet-wise engineering concept from Module 15 (PostgreSQ...

MODULE 16 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

pgvector Hybrid Search, Logical Replication & PostgreSQL DBA/SDE Interviews

Self-explanatory list-and-bullet production mastery module covering pgvector Hybrid Search, Logical Replication & PostgreSQL DBA/SDE Interviews in PostgreSQL Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ Full-Text Search (tsvector / GIN) + pgvector HNSW Embeddings for AI/RAG Workloads — Purpose: Run relational transactions, keyword search, and vector similarity inside a single ACID database. • ✓ Streaming & Logical Replication, WAL Archiving & Point-in-Time Recovery (PITR) — Purpose: Execute zero-data-loss failover drills and major-version upgrades with minimal downtime. • ✓ Reading Complex EXPLAIN (ANALYZE, BUFFERS) Plans & Postgres Interview Scenarios — Purpose: Diagnose Nested Loop vs Hash Join misestimates, spill-to-disk sorts, and lock queues. • ✓ Build, verify, and document 'HNSW pgvector Semantic Search + Full-Text tsvector Hybrid Query Engine' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + Google Gemini AI Technical Interviews.

16.1 Full-Text Search (tsvector / GIN) + pgvector HNSW Embeddings for AI/RAG Workloads [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Full-Text Search (tsvector / GIN) + pgvector HNSW Embeddings for AI/RAG Workloads in real PostgreSQL systems — specifically: Run relational transa...

16.2 Streaming & Logical Replication, WAL Archiving & Point-in-Time Recovery (PITR) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Streaming & Logical Replication, WAL Archiving & Point-in-Time Recovery (PITR) in real PostgreSQL systems — specifically: Execute zero-data-loss f...

16.3 Reading Complex EXPLAIN (ANALYZE, BUFFERS) Plans & Postgres Interview Scenarios [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Reading Complex EXPLAIN (ANALYZE, BUFFERS) Plans & Postgres Interview Scenarios in real PostgreSQL systems — specifically: Diagnose Nested Loop vs...

Module 16 Hands-On Capstone Project (100 Marks): HNSW pgvector Semantic Search + Full-Text tsvector Hybrid Query Engine — Architect, implement, test, and document 'HNSW pgvector Semantic Search + Full-Text tsvector Hybrid Query Engine' applying every bullet-wise engineering concept from Module 16 (pgvector Hybr...

3. LIVE CLIENT PRODUCTION PROJECT, GLOBAL CERTIFICATIONS & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: LedgerPG Multi-Tenant Row-Level Security, Partitioned Time-Series & pgvector Engine • **Confirmed Role:** PostgreSQL 16 Database Architecture & Tuning Intern

Domain & Production Infra: Enterprise SaaS Analytics & AI Search Database Client (PostgreSQL 16 + Declarative Partitioning + Row-Level Security + PgBouncer + pg_stat_statements)

- Architected tenant-isolated PostgreSQL 16 schemas enforced at the database engine via Row-Level Security (RLS) policies.
- Designed B-Tree, GIN (JSONB/Full-Text), partial, and HNSW (pgvector) indexes verified with EXPLAIN (ANALYZE, BUFFERS).
- Implemented monthly declarative table partitioning, CTE analytics, MVCC autovacuum tuning, and PgBouncer pooling.
- Executed lock-safe online migrations and point-in-time recovery drills on a live production PostgreSQL instance.

Aligned Global Vendor Certifications: **EDB-PG16** (EDB Certified PostgreSQL 16 Associate) • **1Z0-909** (Oracle Certified Professional: MySQL 8.0 Database Developer)

Target Job Roles: PostgreSQL Database Engineer, Backend Data & Performance Engineer, Cloud Database Reliability Engineer (DBRE), SQL & Data Platform Specialist

Realistic Annual CTC Scope: **Fresher / Entry-Level Data & SQL Engineer (0–1 Yr):** ₹5.0 LPA – ₹9.2 LPA **Mid-Level PostgreSQL / Backend Engineer (1–3 Yrs):** ₹10.0 LPA – ₹20.0 LPA **Staff Database Architect / Principal DBRE (3–6+ Yrs):** ₹21.0 LPA – ₹42.0+ LPA

MockAttempt Academy Pvt. Ltd. • Email: support@mockattempt.com • Phone/WhatsApp: +91 98765 43210 • Web: https://mockattempt.com/courses/postgresql-training

Try 2 Free Topic-Wise 50-MCQ Mock Tests & Spoken Gemini AI Interviews daily on every topic above before enrolling.