AnovaCloud

FREE PRACTITIONER PACK

SQL Server → PostgreSQL Migration Pack

The constructs automated converters handle — and the ones that need a human.

Pack	SQL Server → PostgreSQL Migration Pack
Document	T-SQL compatibility checklist (5 categories)
Date	2026-09

AnovaCloud — Engineering the AI-Native Enterprise. Frisco, Texas. This pack is a working instrument, not a brochure: complete it with your team, score honestly, and bring it to your discovery call.

HOW TO USE

How to use

Run the feature scan, then walk each category against your codebase. Every High-effort construct becomes a backlog item with an owner and an estimate.

ERROR HANDLING

Error handling

- [ ] **TRY/CATCH → BEGIN/EXCEPTION**
  Rewrite each CATCH block; map ERROR_NUMBER()/ERROR_MESSAGE() to SQLSTATE/SQLERRM.
- [ ] **@@ERROR checks after every statement**
  Replace with GET DIAGNOSTICS after statements that need it.
- [ ] **@@ROWCOUNT**
  Use GET DIAGNOSTICS n = ROW_COUNT.
- [ ] **RAISERROR**
  Rewrite as RAISE EXCEPTION / RAISE NOTICE with matching severity.
- [ ] **XACT_ABORT settings**
  PostgreSQL aborts the whole transaction on error; design savepoints explicitly.

QUERY CONSTRUCTS

Query constructs

- [ ] **TOP (n) → LIMIT**
  Mechanical for simple cases; TOP WITH TIES needs a rewrite.
- [ ] **OFFSET/FETCH paging**
  Syntax is compatible; verify sort stability with ties.
- [ ] **PIVOT / UNPIVOT**
  No equivalent; rewrite with crosstab() or conditional aggregation.
- [ ] **APPLY (CROSS/OUTER) → LATERAL**
  Direct rewrite pattern; test correlated cases.
- [ ] **CTE (WITH)**
  Compatible; note PostgreSQL materializes only when needed (check plans).
- [ ] **Recursive CTE**
  Compatible; SEARCH/CYCLE clauses differ — rewrite cycle detection.
- [ ] **Window functions**
  Largely compatible; check frame defaults and ordered-set aggregates.
- [ ] **OUTPUT clause → RETURNING**
  Rewrite DML OUTPUT as RETURNING; OUTPUT INTO needs CTEs.
- [ ] **MERGE → INSERT ... ON CONFLICT**
  Covers matched-by-target cases only; complex MERGE needs procedural rewrite.

PROCEDURAL CODE

Procedural code

- [ ] **Temp tables (#temp, ##temp)**
  PostgreSQL TEMP TABLE is session-scoped; global ##temp needs redesign.
- [ ] **Table variables (@t)**
  Use temp tables or arrays; no in-memory table-variable equivalent.
- [ ] **Cursors (FAST_FORWARD etc.)**
  Rewrite set-based where possible; bound cursors where not.
- [ ] **Dynamic SQL (sp_executesql)**
  Use EXECUTE ... USING; quote identifiers with quote_ident().
- [ ] **WHILE loops over rowsets**
  Flag for set-based rewrite; loops are the top performance risk.
- [ ] **Scalar UDFs in hot paths**
  Inline or rewrite; per-row function calls are the classic regression.
- [ ] **Multi-statement table-valued functions**
  Rewrite as SQL-language functions or views.
- [ ] **INSTEAD OF / AFTER triggers**
  Syntax compatible; verify per-row vs per-statement firing order.
- [ ] **Nested transaction / savepoint usage**
  PostgreSQL savepoints work; nested COMMIT semantics differ.
- [ ] **Implicit transactions (IMPLICIT_TRANSACTIONS)**
  No equivalent; make transaction boundaries explicit.

FUNCTIONS & EXPRESSIONS

Functions & expressions

- [ ] **GETDATE()/SYSDATETIME() → now()/clock_timestamp()**
  now() is transaction-start; clock_timestamp() is statement time — pick deliberately.
- [ ] **DATEADD/DATEDIFF/DATENAME**
  Use interval arithmetic and date_part()/to_char().
- [ ] **EOMONTH/ DATEFROMPARTS**
  Rewrite with date_trunc() arithmetic.
- [ ] **ISNULL → COALESCE**
  Mechanical; check two-arg vs multi-arg usage.
- [ ] **IIF/CHOOSE → CASE**
  Mechanical rewrite.
- [ ] **TRY_CAST/TRY_CONVERT**
  No direct equivalent; use exception blocks or regex guards.
- [ ] **SUBSTRING (1-based, both)**
  Compatible; verify length semantics on negative start.
- [ ] **CHARINDEX → position()/strpos()**
  Mechanical.
- [ ] **STUFF → overlay()**
  Mechanical.
- [ ] **LEN → length() vs char_length()**
  LEN trims trailing spaces; length() does not — test string comparisons.
- [ ] **String concatenation with NULL**
  SQL Server + yields NULL; PostgreSQL || yields NULL too — but CONCAT() differs. Test.
- [ ] **CASE-sensitive comparisons**
  Decide collation strategy up front; citext or explicit LOWER().

SCHEMA & PROGRAMMABILITY SURFACE

Schema & programmability surface

- [ ] **Schemas (dbo → public mapping)**
  Decide schema naming; three-part names become two-part.
- [ ] **Three/four-part naming**
  Linked-server four-part names need postgres_fdw or redesign.
- [ ] **Indexed views → materialized views**
  No automatic maintenance; schedule REFRESH.
- [ ] **Full-text (CONTAINS/FREETEXT) → tsvector**
  Rebuild catalogs; ranking functions differ.
- [ ] **Sequences (IDENTITY alternative)**
  CACHE/ordering semantics differ; check gap tolerance.
- [ ] **Synonyms**
  Use search_path or views; no direct synonym object.
- [ ] **Partitioned tables/filegroups**
  Declarative partitioning; filegroup layout becomes tablespaces.
- [ ] **In-memory OLTP tables**
  Consider UNLOGGED tables; durability trade-offs must be explicit.
- [ ] **Temporal tables**
  No built-in equivalent; application or trigger-based history.
- [ ] **Graph tables (NODE/EDGE)**
  No equivalent; use Apache AGE or relational redesign.
- [ ] **CLR assemblies**
  No CLR; rewrite in plpythonu/plperl or move logic to the application.
- [ ] **Service Broker**
  No equivalent; replace with LISTEN/NOTIFY or a queue (pgmq, external).
- [ ] **Linked servers**
  postgres_fdw/dblink for PG targets; redesign for heterogeneous sources.
- [ ] **Replication publications**
  Replace with logical replication or Debezium.
- [ ] **Change Tracking/CDC**
  Replace with pgoutput logical decoding / Debezium.