PostgreSQL vs SQL Server dataset benchmark report on same machine

# Purpose and scope

This report compares PostgreSQL and SQL Server on a common retail analytics workload drawn from the dunnhumphy dataset. It covers dataset scale, test method, engine versions and platform, load performance, runtime throughput and latency, CPU and memory observations, and storage footprint. The goal is to explain what each engine delivered under identical conditions and what that means for a production choice. Source notebooks and HTML exports that produced the runs are included in your files.

# Data and environment

The workload used the following tables and row counts: causal\_data 36,786,524 rows, transaction\_data 2,595,732 rows, product 92,353 rows, coupon 124,548 rows, coupon\_redempt 2,318 rows, hh\_demographic 801 rows, campaign\_table 7,208 rows, and campaign\_desc 30 rows. These counts matched across engines.

PostgreSQL version was 16.10 on Debian packaging. SQL Server was SQL Server 2022 Developer Edition on Ubuntu 22.04.5, build 16.0.4210.1 with CU20 GDR applied.

# Method

Data was bulk loaded into each engine and the same timed benchmark was executed. The runtime benchmark used eight concurrent workers for one minute with an 80 percent read ratio. No errors were recorded during either run.

# Results

## Loading and admin time

SQL Server completed the dataset load in 486.21 seconds. PostgreSQL completed in 644.26 seconds. The difference was 158.05 seconds, which means SQL Server’s load phase was about 24.5 percent shorter than PostgreSQL’s load time. Stated the other way, PostgreSQL took about 32.5 percent longer than SQL Server for ingestion on this dataset.

## Throughput and latency under load

PostgreSQL processed 172,258 transactions during its run with measured elapsed time near one minute and achieved 2,779.38 transactions per second. Its 95th percentile latency was 3.173 milliseconds.   
SQL Server processed 155,971 transactions with measured elapsed time at one minute and achieved 2,599.57 transactions per second. Its 95th percentile latency was 3.312 milliseconds.

On these runs PostgreSQL delivered approximately 6.9 percent higher throughput than SQL Server and a slightly lower 95th percentile latency by about 0.139 milliseconds, which is about a 4.2 percent reduction. Those differences held while the workload kept concurrency at eight and a read ratio of 80 percent.

## CPU and memory observations

For SQL Server, average CPU utilization across the sampling window was about 251.5 percent with a 95th percentile at 256.1 percent. Average memory percentage was about 45.8 percent with a similar 95th percentile. That pattern points to modest headroom on memory and a level of CPU use that suggests the server made use of multiple cores without saturating them.

## Storage footprint

Reported database footprint for PostgreSQL was 2,568,770,019 bytes. Reported database footprint for SQL Server was 2,902,458,368 bytes. On those measures PostgreSQL used about 11.5 percent less space overall. Table-level allocations told a more nuanced story. For example, PostgreSQL showed larger figures on the two biggest tables, with causal\_data at about 2.06 to 2.21 GB for PostgreSQL measurements and 1.49 to 1.60 GB for SQL Server measurements when rounded to gigabyte thresholds, while the database total for SQL Server was still higher. The difference between summed table size and total database size was small for PostgreSQL and large for SQL Server, which likely reflects engine-specific overhead such as internal catalogs, indexes, free space reservations, or log file accounting in the measurement call used for SQL Server. These details are from the captured size breakdowns in the result files.

# What the numbers imply

## Load performance vs. runtime performance

If the priority is to refresh or rebuild the warehouse quickly, SQL Server’s load time advantage was clear on this run. It loaded a dataset of more than 36 million causal rows and more than 2.5 million transactions in roughly eight minutes, whereas PostgreSQL took a little under eleven minutes. The gap is meaningful when frequent reloads or large delta ingests are part of the daily cycle.

During live queries under mixed read heavy load, PostgreSQL produced higher throughput and slightly better tail latency. The margin was not extreme but was consistent across the one minute window. For a dashboard or API that values steady low millisecond latency at the 95th percentile, the PostgreSQL result favors that use case. For batch reporting that stresses ingestion speed, the SQL Server result favors that phase.

## Resource use

SQL Server’s CPU profile averaged about two and a half cores busy with peak samples near that level and memory under 50 percent. That profile suggests additional throughput might be available with higher concurrency or larger in-memory working sets.

## Storage tradeoffs

The total database size reported for SQL Server was higher even though the table allocation lines showed some large tables smaller than the PostgreSQL counterparts. This pattern is consistent with engine differences in how database size is measured and how free space or transaction logs are accounted. Teams that operate against strict disk budgets should validate footprint with the same method across engines, then repeat after index creation and after representative vacuum or shrink maintenance to reach steady state. Numbers quoted above come from the run artifacts, not from a normalized external tool, which is important when drawing final conclusions about space use.

# What to choose today

If you need faster initial loads or frequent full reloads, choose SQL Server on the results observed here. If you need higher steady throughput and slightly lower tail latency for a read‑heavy service, choose PostgreSQL on the results observed here. If both matter, run the two follow‑up tracks and choose after tuning. The data collected for this round supports both options depending on the priority.

# Key figures at a glance

PostgreSQL: 172,258 transactions processed, 2,779.38 transactions per second, 95th percentile latency 3.173 ms, load time 644.26 s, reported database size 2,568,770,019 bytes.   
SQL Server: 155,971 transactions processed, 2,599.57 transactions per second, 95th percentile latency 3.312 ms, load time 486.21 s, reported database size 2,902,458,368 bytes, average CPU 251.5 percent and average memory 45.8 percent during the run.