Database Performance Benchmark Explanation

This material explains the performance benchmarking script that measures PostgreSQL and MongoDB CRUD operations. It covers **what the code does**, **what the graphs mean**, and presents both **technical** and **non-technical** perspectives.

---

1. What the Code Does

1. **Connects to databases**
    - PostgreSQL via `psycopg2`
    - MongoDB via `pymongo`
2. **Prepares test tables/collections**
    - PostgreSQL: creates `perf_test` table
    - MongoDB: uses `perf_test` collection
3. **Runs benchmarks** for:
    - **Single CRUD** (Create, Read, Update, Delete) → measures latency of a single operation
    - **Bulk CRUD** (Create, Read, Update, Delete for 10,000 rows/documents) → measures latency and throughput
4. **Calculates metrics**
    - **Average, minimum, and maximum time** per operation (in milliseconds)
    - **Throughput** for bulk operations: how many rows/documents per second
5. **Generates results**
    - **Rich table** in terminal for clear numerical comparison
    - **Automatic conclusions** about which DB is faster for each operation
    - **Matplotlib graphs**:
        - Top panel: average time per operation
        - Bottom panel: bulk throughput

---

2. Interpreting the Graphs

Top Panel: Average Time per Operation (ms)

- **Technical Explanation**
  - Y-axis: time taken per operation in milliseconds
  - X-axis: database operations
  - Bars: skyblue = PostgreSQL, salmon = MongoDB
  - Shows **latency for single and bulk operations**
  - Shorter bars → faster completion → lower latency
  - Critical for **interactive applications** where speed matters per query
- **Non-Technical Explanation**
  - Think of this as a race:
    - Each bar shows how fast a database can do a job
    - Short bar = database completes the task faster
    - Single CRUD: small tasks (like adding a customer)
    - Bulk CRUD: bigger tasks (like adding 10,000 customers at once)
  - You can see which database is faster at completing each task

---

Bottom Panel: Bulk Operations Throughput (rows/documents/sec)

- **Technical Explanation**
  - Y-axis: number of rows/documents processed per second
  - X-axis: bulk operations only
  - Bars: same color scheme
  - Shows **how well each database scales for large data operations**
  - Taller bars = higher throughput → better performance for batch processing
- **Non-Technical Explanation**
  - Imagine a cashier serving customers:
    - Taller bar = more people served per second
    - Shows which database handles a big crowd faster
    - Example: inserting 10,000 records
      - PostgreSQL = 40,000 rows/sec
      - MongoDB = 110,000 rows/sec → MongoDB completes the job faster

---

3. Key Takeaways

Metric	Technical Insight	Layman Insight
Single CRUD latency	Measures micro-performance; lower ms = faster response	Who finishes a small task faster
Bulk CRUD latency	Time taken for large batch operations	How long it takes to do a big job
Bulk throughput	Rows/documents per second; higher = better scaling	Who can handle more work at once
Overall trends	MongoDB generally faster for single inserts and high-volume writes	MongoDB is quicker for small tasks and big crowds; PostgreSQL may be competitive for some bulk reads

---

4. How to Explain the Results

1. **For developers/technical audience:**
    - "The top panel shows per-operation latency (ms), highlighting MongoDB's faster single inserts. The bottom panel shows bulk throughput; MongoDB outperforms PostgreSQL in bulk writes."
2. **For non-technical audience:**
    - "The top graph shows who can finish a task faster (shorter bars = faster). The bottom graph shows who can handle a crowd better (taller bars = more work done per second). MongoDB is faster both for small tasks and big jobs."

---

5. Why This Matters

- **Latency (time per operation)** → affects user experience in real-time apps
- **Throughput (bulk operations)** → affects performance for analytics, migrations, and batch jobs
- Understanding both helps in **choosing the right database** for your workload.
