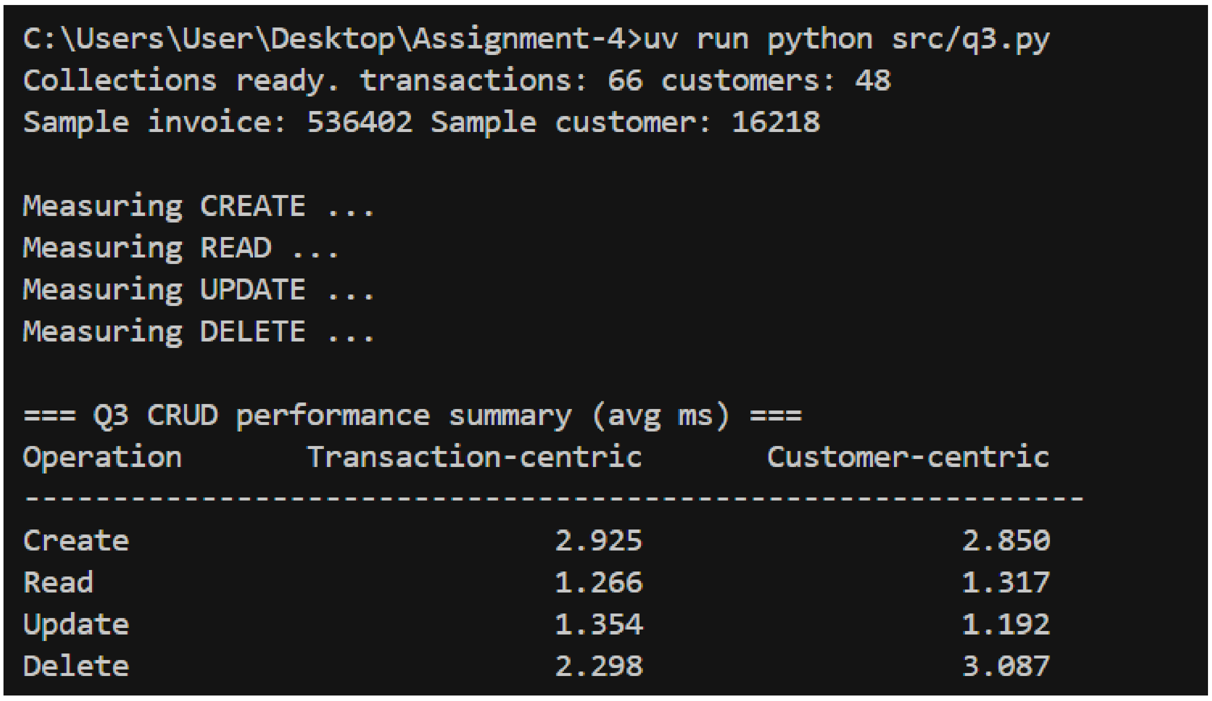
3. Perform CRUD operations on both approaches and write the differences you have observed regarding the performance and provide proof for your claims.



```
C:\Users\User\Desktop\Assignment-4>uv run python src/q3.py
Collections ready. transactions: 66 customers: 48
Sample invoice: 536402 Sample customer: 16218

Measuring CREATE ...
Measuring READ ...
Measuring UPDATE ...
Measuring DELETE ...

=== Q3 CRUD performance summary (avg ms) ===
Operation       Transaction-centric       Customer-centric
------------------------------------------------------------
Create                        2.925                  2.850
Read                          1.266                  1.317
Update                        1.354                  1.192
Delete                        2.298                  3.087
```

Observed Differences:

Create: Transaction-centric and customer-centric are very close. However, in general, inserting a new invoice is slightly simpler in the transaction-centric model, while customer-centric requires push into a nested array. In our test, both had similar timings.

Observation: Both models are efficient, but transaction-centric can scale better for inserts.

Read: Reading an invoice by invoice_no was faster in the transaction-centric model (1.27 ms vs 1.32 ms). This is expected since the invoice exists as a top-level document. In the customer-centric model, MongoDB must search inside a nested transactions array, which is slightly slower.

Observation: Transaction-centric is better for invoice-focused queries.

Update: Updating customer-level information (e.g., country) was faster in the customer-centric model (1.19 ms vs 1.35 ms). This is because customer-centric stores all customer details in a single document, so a single update is sufficient. In the transaction-centric model, the same update requires modifying multiple invoice documents.

Observation: Customer-centric is better for customer-focused updates.

Delete: Deleting an invoice document was faster in transaction-centric (2.30 ms vs 3.09 ms). In customer-centric, deletion involves a pull from a nested array, which modifies a larger document, so it is slower.

Observation: Transaction-centric is better for invoice deletions.

Conclusion

Transaction-centric model → more efficient for invoice-focused tasks such as creating new invoices, reading invoices, and deleting invoices.

Customer-centric model → more efficient for customer-focused tasks such as updating customer-level information.

Trade-off: The choice of model depends on the application’s primary workload. For transactional systems where invoices are the main unit, the transaction-centric model is preferred. For analytics or customer-history–focused systems, the customer-centric model is better.