Software System Development

Lab Activity 3
21 August

Lecture: NoSQL

Database: `sample_supplies` **Collection**: `sales`

Total Marks: 20 **Deadline**: 22 August, 5 PM

Instructions

- Use MongoDB Shell `mongosh` only. No scripts and no code files.
- Atlas option: load Sample Data to get `sample_supplies.sales`. Local option: import the JSON mirror below. Do not write to `sales`.
- First clone `sales` into `sales_work`. All writes go to `sales_work`. Reads in Tasks 5 to 10 must use `sales`.
- For each question k run exactly one command in `mongosh`. Copy that command into `qk_query.txt` and the exact one line JSON output into `qk_output.txt`.
- Always wrap your expression with `printjsononeline(...)` so the output is a single JSON value on one line.
- Work individually. Plagiarism results in zero.

Dataset setup

Atlas: open your cluster, click Collections, then Load Sample Dataset. Use `sample_supplies.sales`.

Local: download JSON and import

```
curl -L -o sales.json \
  https://raw.githubusercontent.com/neelabalan/mongodb-sample-dataset/main/
    sample_supplies/sales.json
mongoimport --uri "mongodb://localhost:27017" \
  --db sample_supplies --collection sales \
  --file sales.json --jsonArray
```

How to run in mongosh

```
mongosh
use sample_supplies
```

Clone base to a working copy once:

```
printjsononeline((function(){
  db.sales_work?.drop();
  db.sales.aggregate([
    {$match:{}},
    {$merge:{into:"sales_work", whenMatched:"fail", whenNotMatched:"insert"}}
  ]);
  return {src: db.sales.countDocuments({}), work: db.sales_work.countDocuments
    ({})};
})())
```

Patterns you will reuse:

```
# Array from find
printjsononeline(
  db.sales.find({FILTER}, {PROJ}).sort({SORT}).limit(N).toArray()
)
# Array from aggregation
printjsononeline(
  db.sales.aggregate([ STAGES... ]).toArray()
)
# Single object or number
printjsononeline({count: db.sales.countDocuments({FILTER})})
```

Submission format

Create a folder `<roll_number>_lab3` and submit it as a zip file. For each question k add:

- `qk_query.txt` one line containing the exact command you executed.
- `qk_output.txt` one line containing the exact JSON printed by `mongosh`.

Example:

```
# q1_query.txt
printjsononeline({count: db.sales.countDocuments({})})

# q1_output.txt
{"count":16050}
```

Write any assumptions or interpretations in a `README.txt`.

Context

You are a new data analyst at *SupplyChainX*, a global office supplies retailer. The ops lead wants quick changes in a sandbox copy and fast readouts from the live data.

Tasks

Ten tasks at 2 marks each. Tasks 1 to 4 target `sales_work`. Tasks 5 to 10 target `sales`. Numeric checks use absolute tolerance `1e-6`.

Task 1. Create [2 Marks]

Ops lead asked you to record a training order so downstream tools see a fresh shape.
Insert one document into `sales_work` with your details:

- `orderId`: `LAB_<ROLL>`
- `saleDate`: ISO date of today
- `items`: one element with `name` of your choice, tags `["lab","custom"]`, `price: 10.0`, `quantity: 1`
- `storeLocation`: `"Training"`
- `customer`: set `email` to your email, `age` to your age, `satisfaction` to 4
- `couponUsed`: `false`, `purchaseMethod`: `"Online"`

Output: insert result object with `acknowledged` and `insertedId`.

Task 2. Read [2 Marks]

Review team wants to confirm the row landed with the right keys.
From `sales_work`, fetch your document by `orderId`. Project `orderId`, `storeLocation`, `purchaseMethod`, and first item name as `item0`. Exclude `_id`.
Output: a single object.

Task 3. Update [2 Marks]

Review team flagged that you undercounted quantities.
On `sales_work`, for your `orderId` increment `items.0.quantity` by 2 and set `customer.satisfaction` to 5.
Output: update result object with `matchedCount` and `modifiedCount`.

Task 4. Delete [2 Marks]

Sandbox cleanup request.
Delete your inserted row from `sales_work` by `orderId`.
Output: delete result object with `deletedCount`.

Task 5. Top stores by revenue [2 Marks]

The ops lead wants a quick league table of stores.
On `sales`, compute revenue per `storeLocation` as sum of `items.price * items.quantity`. Return top 5 sorted by `revenue` desc then `storeLocation` asc. Convert revenue to a JSON number.
Output: array {`storeLocation`, `revenue`}.

Task 6. Monthly revenue, 2015 [2 Marks]

Finance wants a 2015 month series for reconciliation.

On `sales`, for orders in 2015 return {`month: "YYYY-MM"`, `revenue`} sorted by `month` asc. Only months present in data should appear. Convert revenue to a JSON number.

Output: array of objects.

Task 7. Basket size by channel [2 Marks]

Product Manager suspects phone orders contain fewer distinct items.

On `sales`, for each `purchaseMethod` compute average number of distinct item names per order. Round to 2 decimals. Sort by `avg_distinct_items` desc then `purchaseMethod` asc.

Output: array {`purchaseMethod`, `avg_distinct_items`}.

Task 8. Tags dictionary [2 Marks]

Marketing is cleaning tag taxonomy.

On `sales`, list all distinct `items.tags` as a sorted array of strings.

Output: array of strings sorted asc.

Task 9. Payment channel quality proxy [2 Marks]

Support asks which channel yields happier customers.

On `sales`, using `customer.satisfaction` where present, return average satisfaction per `purchaseMethod` for orders since 2015. Round to 2 decimals. Sort by `avg_satisfaction` desc then `purchaseMethod` asc.

Output: array {`purchaseMethod`, `avg_satisfaction`, `n`}.

Task 10. Top items by revenue [2 Marks]

Buying wants the revenue leaders.

On `sales`, return the top 10 item names by total revenue sum of `price * quantity`. Convert revenue to a JSON number. Sort by `revenue` desc then `name` asc.

Output: array {`name`, `revenue`}.

Example formatting

Task 5 example:

```
# q5_query.txt
printjsononeline(db.sales.aggregate([
  {$unwind:"$items"},
  {$group:{_id:"$storeLocation",
           rev:{$sum:{$multiply:["$items.price","$items.quantity"]}}}},
  {$project:{_id:0,storeLocation:"$_id",revenue:{$toDouble:"$rev"}}},
  {$sort:{revenue:-1,storeLocation:1}},
  {$limit:5}
]).toArray())

# q5_output.txt
[{"storeLocation":"...","revenue":12345.67}, ...]
```