

## GroupB-3.

Aim: Implement aggregation and indexing with suitable example using MongoDB Problem statement: Create a collection named "ORDERS" that contain documents of the following prototype and solve the following queries: { cust\_id: "abc123", ord\_date: new Date("Oct 04, 2012"), status: 'A', price: 50, items: [ { sku: "xxx", qty: 25, price: 1 }, { sku: "yyy", qty: 25, price: 1 } ] }

### a. Count all records from orders

```
Command Prompt - mongo
local 0.000GB
test 0.000GB
> show collections
Book
Orders
book
> db.Orders.insertOne({cust_id:"abc123",ord_date:new Date("Oct 04,2012"),status:'A',price:50,items:[{sku:"xxx",qty:25,price:1},{sku:"yyy",qty:25,price:1}]}
... )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60782856063a8b94c80cf55d")
}
> db.Orders.count()
1
>
```

### b. Sum the price field from orders

```
Command Prompt - mongo
> show collections
Book
Orders
book
> db.Orders.insertOne({cust_id:"abc123",ord_date:new Date("Oct 04,2012"),status:'A',price:50,items:[{sku:"xxx",qty:25,price:1},{sku:"yyy",qty:25,price:1}]}
... )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60782856063a8b94c80cf55d")
}
> db.Orders.count()
1
> db.Orders.aggregate([{$group:{_id:$by_user,price:{$sum:$likes}}}]
uncaught exception: ReferenceError: $by_user is not defined :
@ (shell):1:31
> db.Orders.aggregate([{$group:{_id:"$by_user",price:{$sum:$likes}}}]
{ "_id" : null, "price" : 0 }
>
```

c. For each unique cust\_id, sum the price field.

```
Command Prompt - mongo
> db.Orders.aggregate([{$group:{_id:"$by_user",price:{$sum:"$likes"}}}])
{ "_id" : null, "price" : 0 }
>
```

d. For each unique cust\_id, sum the price field, results sorted by sum.

```
Command Prompt - mongo
> db.Orders.aggregate([{$group:{_id:"$by_user",price:{$sum:1}}}])
{ "_id" : null, "price" : 2 }
>
```

e. For each unique cust\_id, ord\_date grouping, sum the price field.

```
Command Prompt - mongo
@ (shell):1:31
> db.Orders.aggregate([{$group:{_id:"$by_user",price:{$sum:"$likes"}}}])
{ "_id" : null, "price" : 0 }
> db.Orders.insertOne({cust_id:"abc124",ord_date:new Date("Oct 12,2012"),status:'A',price:60,items:[{sku:"xxx",qty:25,price:2},{sku:"yyy",qty:25,price:2}] })
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60782a36063a8b94c80cf55e")
}
> db.Orders.count()
2
> db.Orders.aggregate([{$group:{_id:"$by_user",price:{$sum:"$likes"}}}])
{ "_id" : null, "price" : 0 }
> db.Orders.aggregate([{$group:{_id:"$by_user",price:{$sum:"$likes"}}}]).sort({sum:1})
uncaught exception: TypeError: db.Orders.aggregate(...).sort is not a function :
@ (shell):1:1
> db.Orders.aggregate([{$group:{_id:"$by_user",price:{$sum:"$likes"}}}]).sort({sum:1})
uncaught exception: TypeError: db.Orders.aggregate(...).sort is not a function :
@ (shell):1:1
> db.Orders.aggregate([{$group:{_id:"$by_user",price:{$sum:1}}}])
{ "_id" : null, "price" : 2 }
> db.Orders.aggregate([{$group:{_id:"$by_user",_id:"$by_user",price:{$sum:1}}}])
{ "_id" : null, "price" : 2 }
```

f. For cust\_id with multiple records, return the cust\_id and the corresponding record count.

```
Command Prompt - mongo
> db.Orders.insertOne({cust_id:"abc124",ord_date:new Date("Oct 12,2012"),status:'A',price:60,items:[{sku:"xxx",qty:25,price:2},{sku:"yyy",qty:25,price:2}]} )
> db.Orders.insertOne({cust_id:"pqr258",ord_date:new Date("Oct 16,2012"),status:'A',price:60,items:[{sku:"xxx",qty:25,price:2},{sku:"yyy",qty:25,price:2}]} )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60782cbe063a8b94c80cf55f")
}
> db.Orders.insertOne({cust_id:"xyz896",ord_date:new Date("Nov 16,2012"),status:'A',price:60,items:[{sku:"xxx",qty:25,price:2},{sku:"yyy",qty:25,price:2}]} )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60782cdc063a8b94c80cf560")
}
> db.Orders.count()
4
>
```

g. For each unique cust\_id, ord\_date grouping, sum the price field and return only where the sum is greater than 250.

h. For each unique cust\_id with status A, sum the price field.

```
Command Prompt - mongo
> db.Orders.aggregate([{$group:{_id:"$by_user",status:{$sum:1}}}] )
{ "_id" : null, "status" : 4 }
>
```