```
1    DBMSL Assignment No 10
2    Name: Himanshu Lonkar
3    Roll No:C31149
4    Batch:T3
5
6    Output:
7
8        #Data
9
10           Orders> db.order.find()
11               [
12                 {
13                   _id: ObjectId("652e5ba74b17d3480662a1bb"),
14                   order_id: 1,
15                   cust_id: 'A1',
16                   cust_name: 'ABC',
17                   phone_no: [ 1234567890, 987654321 ],
18                   email: 'ABC@gmail.com',
19                   item: 'Laptop',
20                   DOR: ISODate("2022-12-31T18:30:00.000Z"),
21                   qty: 2,
22                   amt: 10000,
23                   status: 'P'
24                 },
25                 {
26                   _id: ObjectId("652e5c574b17d3480662a1bc"),
27                   order_id: 2,
28                   cust_id: 'A2',
29                   cust_name: 'XYZ',
30                   phone_no: 1234567890,
31                   email: 'XYZ@gmail.com',
32                   item: 'TV',
33                   DOR: ISODate("2023-02-12T18:30:00.000Z"),
34                   qty: 1,
35                   amt: 20000,
36                   status: 'D'
37                 }
38               ]
39
40       1.  A) Create a simple index on cust_id and also create a simple index on Item_name.
41
42               Orders> db.order.ensureIndex({'cust_id':1})
43                   [ 'cust_id_1' ]
44               Orders> db.order.ensureIndex({'item':1})
45                   [ 'item_1' ]
46               Orders> db.order.getIndexes()
47                   [
48                     { v: 2, key: { _id: 1 }, name: '_id_' },
49                     { v: 2, key: { cust_id: 1 }, name: 'cust_id_1' },
50                     { v: 2, key: { item: 1 }, name: 'item_1' }
51                   ]
52
53          B)Try to make a duplicate entry.
54
55               Orders>
               db.order.insertOne({order_id:2,cust_id:'A2',cust_name:'XYZ',phone_no:12345678
               90,email:'XYZ@gmail.com',item:'TV',DOR:new
               Date('2023-2-13'),qty:2,amt:20000,status:'P'})
56                   {
57                     acknowledged: true,
58                     insertedId: ObjectId("652e60d44b17d3480662a1bd")
59                   }
60
61               Orders> db.order.find({cust_id:'A2'})
62                   [
63                     {
64                       _id: ObjectId("652e5c574b17d3480662a1bc"),
65                       order_id: 2,
66                       cust_id: 'A2',
```

```
67                              cust_name: 'XYZ',
68                              phone_no: 1234567890,
69                              email: 'XYZ@gmail.com',
70                              item: 'TV',
71                              DOR: ISODate("2023-02-12T18:30:00.000Z"),
72                              qty: 1,
73                              amt: 20000,
74                              status: 'D'
75                          },
76                          {
77                              _id: ObjectId("652e60d44b17d3480662a1bd"),
78                              order_id: 2,
79                              cust_id: 'A2',
80                              cust_name: 'XYZ',
81                              phone_no: 1234567890,
82                              email: 'XYZ@gmail.com',
83                              item: 'TV',
84                              DOR: ISODate("2023-02-12T18:30:00.000Z"),
85                              qty: 2,
86                              amt: 20000,
87                              status: 'P'
88                          }
89                      ]
90
91      2.   A)Create unique index on the order_id key.
92
93              Orders> db.order.ensureIndex({order_id:1},{unique:1})
94                  [ 'order_id_1' ]
95              Orders> db.order.getIndexes()
96                  [
97                      { v: 2, key: { _id: 1 }, name: '_id_' },
98                      { v: 2, key: { cust_id: 1 }, name: 'cust_id_1' },
99                      { v: 2, key: { item: 1 }, name: 'item_1' },
100                     { v: 2, key: { order_id: 1 }, name: 'order_id_1', unique: true }
101                 ]
102         B)Try to make duplicate entry.
103
104             Orders>
             db.order.insertOne({order_id:2,cust_id:'A2',cust_name:'XYZ',phone_no:12345678
             90,email:'XYZ@gmail.com',item:'TV',DOR:new
             Date('2023-2-13'),qty:2,amt:20000,status:'P'})
105
106             MongoServerError: E11000 duplicate key error collection: Orders.order index:
             order_id_1 dup key: { order_id: 2 }
107
108     3.   A)Create a multikey index on phone_no.
109
110             Orders> db.order.ensureIndex({phone_no:1},{multi:1})
111                 [ 'phone_no_1' ]
112             Orders> db.order.getIndexes()
113                 [
114                     { v: 2, key: { _id: 1 }, name: '_id_' },
115                     { v: 2, key: { cust_id: 1 }, name: 'cust_id_1' },
116                     { v: 2, key: { item: 1 }, name: 'item_1' },
117                     { v: 2, key: { order_id: 1 }, name: 'order_id_1', unique: true },
118                     { v: 2, key: { phone_no: 1 }, name: 'phone_no_1' }
119                 ]
120
121         B)Find the customers with 2 phone nos.
122
123             Orders> db.order.find({phone_no:{$size:2}})
124                 [
125                     {
126                         _id: ObjectId("652e5ba74b17d3480662a1bb"),
127                         order_id: 1,
128                         cust_id: 'A1',
129                         cust_name: 'ABC',
130                         phone_no: [ 1234567890, 987654321 ],
131                         email: 'ABC@gmail.com',
```

```
132                          item: 'Laptop',
133                          DOR: ISODate("2022-12-31T18:30:00.000Z"),
134                          qty: 2,
135                          amt: 10000,
136                          status: 'P'
137                        }
138                    ]
139    4.   A)Create a sparse index on email_id key and show the effects without indexing.
140
141          Order> db.order.find({email:'ABC@gmail.com'}).explain()
142              {
143                explainVersion: '2',
144                queryPlanner: {
145                  namespace: 'Order.order',
146                  indexFilterSet: false,
147                  parsedQuery: { email: { '$eq': 'ABC@gmail.com' } },
148                  queryHash: 'B9CE814D',
149                  planCacheKey: 'A757FE25',
150                  maxIndexedOrSolutionsReached: false,
151                  maxIndexedAndSolutionsReached: false,
152                  maxScansToExplodeReached: false,
153                  winningPlan: {
154                    queryPlan: {
155                      stage: 'COLLSCAN',
156                      planNodeId: 1,
157                      filter: { email: { '$eq': 'ABC@gmail.com' } },
158                      direction: 'forward'
159                    },
160                    slotBasedPlan: {
161                      slots: '$$RESULT=s5 env: { s3 = 1697544085775 (NOW), s7 =
                         "ABC@gmail.com", s1 =
                         TimeZoneDatabase(Africa/Khartoum...America/Indiana/Knox)
                         (timeZoneDB), s2 = Nothing (SEARCH_META) }',
162                      stages: '[1] filter {traverseF(s4, lambda(l1.0) { ((l1.0 == s7)
                         ?: false) }, false)} \n' +
163                        '[1] scan s5 s6 none none none none lowPriority [s4 = email]
                         @"e58977ea-171d-48eb-8876-6e86f61146ad" true false '
164                    }
165                  },
166                  rejectedPlans: []
167                },
168                command: { find: 'order', filter: { email: 'ABC@gmail.com' }, '$db':
                   'Order' },
169                serverInfo: {
170                  host: 'DESKTOP-DC8IT0V',
171                  port: 27017,
172                  version: '7.0.2',
173                  gitVersion: '02b3c655e1302209ef046da6ba3ef6749dd0b62a'
174                },
175                serverParameters: {
176                  internalQueryFacetBufferSizeBytes: 104857600,
177                  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
178                  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
179                  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
180                  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
181                  internalQueryProhibitBlockingMergeOnMongoS: 0,
182                  internalQueryMaxAddToSetBytes: 104857600,
183                  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
184                  internalQueryFrameworkControl: 'trySbeEngine'
185                },
186                ok: 1
187              }
188    B)Create the sparse index.
189
190          Orders> db.order.ensureIndex({email:1},{sparse:1})
191              [ 'email_1' ]
192          Orders> db.order.getIndexes()
193              [
194                { v: 2, key: { _id: 1 }, name: '_id_' },
```

```
195                          { v: 2, key: { cust_id: 1 }, name: 'cust_id_1' },
196                          { v: 2, key: { item: 1 }, name: 'item_1' },
197                          { v: 2, key: { order_id: 1 }, name: 'order_id_1', unique: true },
198                          { v: 2, key: { phone_no: 1 }, name: 'phone_no_1' },
199                          { v: 2, key: { email: 1 }, name: 'email_1', sparse: true }
200                      ]
201
202          C)show the effects with indexing.
203
204              Orders> db.order.find({email:'ABC@gmail.com'}).explain()
205                  {
206                      explainVersion: '2',
207                      queryPlanner: {
208                        namespace: 'Orders.order',
209                        indexFilterSet: false,
210                        parsedQuery: { email: { '$eq': 'ABC@gmail.com' } },
211                        queryHash: 'B9CE814D',
212                        planCacheKey: '674E89D1',
213                        maxIndexedOrSolutionsReached: false,
214                        maxIndexedAndSolutionsReached: false,
215                        maxScansToExplodeReached: false,
216                        winningPlan: {
217                          queryPlan: {
218                            stage: 'FETCH',
219                            planNodeId: 2,
220                            inputStage: {
221                              stage: 'IXSCAN',
222                              planNodeId: 1,
223                              keyPattern: { email: 1 },
224                              indexName: 'email_1',
225                              isMultiKey: false,
226                              multiKeyPaths: { email: [] },
227                              isUnique: false,
228                              isSparse: true,
229                              isPartial: false,
230                              indexVersion: 2,
231                              direction: 'forward',
232                              indexBounds: { email: [ '["ABC@gmail.com", "ABC@gmail.com"]' ]
                                  }
233                            }
234                          },
235                          slotBasedPlan: {
236                            slots: '$$RESULT=s11 env: { s3 = 1697544343321 (NOW), s6 =
                              KS(3C41424340676D61696C2E636F6D00FE04), s1 =
                              TimeZoneDatabase(Africa/Khartoum...America/Indiana/Knox)
                              (timeZoneDB), s2 = Nothing (SEARCH_META), s5 =
                              KS(3C41424340676D61696C2E636F6D000104), s10 = {"email" : 1} }',
237                            stages: '[2] nlj inner [] [s4, s7, s8, s9, s10] \n' +
238                              '    left \n' +
239                              '        [1] cfilter {(exists(s5) && exists(s6))} \n' +
240                              '        [1] ixseek s5 s6 s9 s4 s7 s8 []
                              @"79cf34d4-e449-4210-a944-1b2e36eb936a" @"email_1" true \n' +
241                              '    right \n' +
242                              '        [2] limit 1 \n' +
243                              '        [2] seek s4 s11 s12 s7 s8 s9 s10 []
                              @"79cf34d4-e449-4210-a944-1b2e36eb936a" true false \n'
244                          }
245                        },
246                        rejectedPlans: []
247                      },
248                      command: {
249                        find: 'order',
250                        filter: { email: 'ABC@gmail.com' },
251                        '$db': 'Orders'
252                      },
253                      serverInfo: {
254                        host: 'DESKTOP-DC8IT0V',
255                        port: 27017,
256                        version: '7.0.2',
```

```
257                             gitVersion: '02b3c655e1302209ef046da6ba3ef6749dd0b62a'
258                         },
259                         serverParameters: {
260                             internalQueryFacetBufferSizeBytes: 104857600,
261                             internalQueryFacetMaxOutputDocSizeBytes: 104857600,
262                             internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
263                             internalDocumentSourceGroupMaxMemoryBytes: 104857600,
264                             internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
265                             internalQueryProhibitBlockingMergeOnMongoS: 0,
266                             internalQueryMaxAddToSetBytes: 104857600,
267                             internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
268                             internalQueryFrameworkControl: 'trySbeEngine'
269                         },
270                         ok: 1
271                     }
272         5.   A)Display all indexes created on order collection.
273
274                 Orders> db.order.getIndexes()
275                     [
276                         { v: 2, key: { _id: 1 }, name: '_id_' },
277                         { v: 2, key: { cust_id: 1 }, name: 'cust_id_1' },
278                         { v: 2, key: { item: 1 }, name: 'item_1' },
279                         { v: 2, key: { order_id: 1 }, name: 'order_id_1', unique: true },
280                         { v: 2, key: { phone_no: 1 }, name: 'phone_no_1' },
281                         { v: 2, key: { email: 1 }, name: 'email_1', sparse: true }
282                     ]
283
284         B)Also show the size of indexes.
285
286             Orders> db.order.totalIndexSize()
287                 188416
288
289     6.Delete all indexes.
290
291         Orders> db.order.dropIndexes()
292             {
293               nIndexesWas: 6,
294               msg: 'non-_id indexes dropped for collection',
295               ok: 1
296             }
297         Orders> db.order.getIndexes()
298         [ { v: 2, key: { _id: 1 }, name: '_id_' } ]
299
300     7.   A) Find Total no of orders received so far
301
302             Orders> db.order.find().count()
303                 3
304
305         B) How many orders are pending.
306
307             Orders> db.order.find({status:'P'}).count()
308                 2
309
310    #Inserting Data
311        Orders>
        db.order.insertOne({order_id:4,cust_id:'A4',cust_name:'LMN',phone_no:[1234567890.2345
        678910],item:'Microwave',DOR:new Date('2023-3-20'),qty:6,amt:20000,status:'D'})
312            {
313              acknowledged: true,
314              insertedId: ObjectId("652e7b88afcf0cbe9e7187a8")
315            }
316        Orders>
        db.order.insertOne({order_id:5,cust_id:'A4',cust_name:'LMN',phone_no:[1234567890.2345
        678910],item:'TV',DOR:new Date('2023-4-20'),qty:4,amt:50000,status:'D'})
317            {
318              acknowledged: true,
319              insertedId: ObjectId("652e7bdaafcf0cbe9e7187a9")
320            }
321
```

```
  322        8.  Display all customer names of orders collection with no repetition.

  323

  324              Orders> db.order.distinct('cust_name')
  325                  [ 'ABC', 'LMN', 'XYZ' ]

  326

  327       9.Show results and details of sorting documents based on amount.

  328

  329          Orders> db.order.find().sort({'amt':1})
  330              [
  331                  {
  332                    _id: ObjectId("652e5ba74b17d3480662a1bb"),
  333                    order_id: 1,
  334                    cust_id: 'A1',
  335                    cust_name: 'ABC',
  336                    phone_no: [ 1234567890, 987654321 ],
  337                    email: 'ABC@gmail.com',
  338                    item: 'Laptop',
  339                    DOR: ISODate("2022-12-31T18:30:00.000Z"),
  340                    qty: 2,
  341                    amt: 10000,
  342                    status: 'P'
  343                  },
  344                  {
  345                    _id: ObjectId("652e5c574b17d3480662a1bc"),
  346                    order_id: 3,
  347                    cust_id: 'A2',
  348                    cust_name: 'XYZ',
  349                    phone_no: 1234567890,
  350                    email: 'XYZ@gmail.com',
  351                    item: 'TV',
  352                    DOR: ISODate("2023-02-12T18:30:00.000Z"),
  353                    qty: 1,
  354                    amt: 20000,
  355                    status: 'D'
  356                  },
  357                  {
  358                    _id: ObjectId("652e60d44b17d3480662a1bd"),
  359                    order_id: 2,
  360                    cust_id: 'A2',
  361                    cust_name: 'XYZ',
  362                    phone_no: 1234567890,
  363                    email: 'XYZ@gmail.com',
  364                    item: 'TV',
  365                    DOR: ISODate("2023-02-12T18:30:00.000Z"),
  366                    qty: 2,
  367                    amt: 20000,
  368                    status: 'P'
  369                  },
  370                  {
  371                    _id: ObjectId("652e7b88afcf0cbe9e7187a8"),
  372                    order_id: 4,
  373                    cust_id: 'A4',
  374                    cust_name: 'LMN',
  375                    phone_no: [ 1234567890.2345679 ],
  376                    item: 'Microwave',
  377                    DOR: ISODate("2023-03-19T18:30:00.000Z"),
  378                    qty: 6,
  379                    amt: 20000,
  380                    status: 'D'
  381                  },
  382                  {
  383                    _id: ObjectId("652e7bdaafcf0cbe9e7187a9"),
  384                    order_id: 5,
  385                    cust_id: 'A4',
  386                    cust_name: 'LMN',
  387                    phone_no: [ 1234567890.2345679 ],
  388                    item: 'TV',
  389                    DOR: ISODate("2023-04-19T18:30:00.000Z"),
  390                    qty: 4,
```

```
                    amt: 50000,
                    status: 'D'
                }
            ]

10.Show how many orders are placed by each customer.

    Orders> db.order.aggregate({$group:{_id:'$cust_name',Count:{$sum:1}}})
        [
          { _id: 'XYZ', Count: 2 },
          { _id: 'ABC', Count: 1 },
          { _id: 'LMN', Count: 2 }
        ]

11.Display all customer ids and their total pending order amount in descending
order.

    Orders>
    db.order.aggregate({$match:{status:'P'}},{$group:{_id:'$cust_id',sum_p:{$sum:'$am
    t'}}},{$sort:{sum_p:-1}},{$limit:1})
        [ { _id: 'A2', sum_p: 20000 } ]

12.Display all customer ids in ascending order with total order amount which have
been is delivered.

    Orders>
    db.order.aggregate({$match:{status:'D'}},{$group:{_id:'$cust_id',sum_p:{$sum:'$am
    t'}}},{$sort:{sum_p:1}})
        [ { _id: 'A2', sum_p: 20000 }, { _id: 'A4', sum_p: 70000 } ]

13.Show top three Selling Items from orders collection.

    Orders>
    db.order.aggregate({$group:{_id:'$item',sum_item:{$sum:'$qty'}}},{$sort:{sum_item
    :-1}},{$limit:3})
        [
          { _id: 'TV', sum_item: 7 },
          { _id: 'Microwave', sum_item: 6 },
          { _id: 'Laptop', sum_item: 2 }
        ]

14.Find the date on which maximum orders are received.

    Orders>
    db.order.aggregate({$group:{_id:'$DOR',count_order:{$sum:1}}},{$sort:{count_order
    :-1}},{$limit:1})
        [ { _id: ISODate("2023-02-12T18:30:00.000Z"), count_order: 2 } ]

15.Find which customer has placed maximum orders.

    Orders>
    db.order.aggregate({$group:{_id:'$cust_name',count_orderid:{$sum:1}}},{$sort:{cou
    nt_orderid:-1}},{$limit:1})
        [ { _id: 'XYZ', count_orderid: 2 } ]
```