```
> use vit
< switched to db vit
> db.sales2.aggregate([{$group:{_id:"$item",maxquantity:{$max:"$quantity"}}}])
<
> db.sales.aggregate([{$group:{_id:"$item",maxquantity:{$max:"$quantity"}}}])
< {
    _id: 'Lattes',
    maxquantity: 30
  }
  {
    _id: 'Mochas',
    maxquantity: 11
  }
  {
    _id: 'Cappuccino',
    maxquantity: 20
  }
  {
    _id: 'Americanos',
    maxquantity: 22
  }
> db.sales.aggregate([
```

```
> db.sales.aggregate([
    {
        $addFields: {
            totalValue: { $multiply: ["$price", "$quantity"] }
        }
    },
    {
        $group: {
            _id: null,
            maxTotalValue: { $max: "$totalValue" }
        }
    }
])
< {
  _id: null,
  maxTotalValue: 750
}
> db.sales.aggregate([
    { $addFields: { totalValue: { $multiply: ["$price", "$quantity"] } } },
    { $sort: { totalValue: -1 } },
    { $limit: 1 },
    { $project: { item: 1, totalValue: 1, _id: 0 } }
])
< {
  item: 'Lattes',
  totalValue: 750
}
```

```
> db.sales.aggregate([
    {
        $addFields: {
            sortedValues: { $cond: { if: { $gt: ["$price", "$quantity"] }, then: ["$price", "$quantity"], else: ["$
        }
    },
    {
        $addFields: {
            secondMaxValue: { $arrayElemAt: ["$sortedValues", 1] }
        }
    },
    {
        $project: { item: 1, secondMaxValue: 1, _id: 0 }
    }
])
< {
  item: 'Americanos',
  secondMaxValue: 5
}
{
  item: 'Cappuccino',
  secondMaxValue: 6
}
{
  item: 'Lattes',
  secondMaxValue: 15
}
{
```

```
> db.sales.aggregate([
    {
      $setWindowFields: {
        partitionBy: "$item",              // Group by item
        sortBy: { quantity: -1 },          // Sort by quantity descending
        output: {
          rank: { $rank: {} }              // Assign rank within each group
        }
      }
    },
    {
      $match: { rank: 2 }                  // Keep only documents with rank 2 (2nd highest quantity per item)
    },
    {
      $project: {
        _id: 0,
        item: 1,
        price: 1,
        size: 1,
        quantity: 1,
        rank: 1                           // Optional: include rank for debugging/visibility
      }
    }
  ])
< {
    item: 'Americanos',
    price: 10,
    size: 'Grande',
```

```
> db.sales.createIndex({item:1})
< item_1
> db.sales.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { item: 1 }, name: 'item_1' }
  ]
> db.sales.dropIndex({_id})
✖ ▸ ReferenceError: _id is not defined
> db.users.insertMany([
     { email:  "john@test.com", name: "john"},
     { email:  "jane@test.com", name: "jane"},
  ]);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('6841a1a1c14e48208d9dcc1f'),
      '1': ObjectId('6841a1a1c14e48208d9dcc20')
    }
  }
> db.users.find()
< {
    _id: ObjectId('6841a1a1c14e48208d9dcc1f'),
    email: 'john@test.com',
    name: 'john'
  }
```

```
> db.tempProducts.find({ category: "tablet" }).explain("executionStats")
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'vit.tempProducts',
      parsedQuery: {
        category: {
          '$eq': 'tablet'
        }
      },
      indexFilterSet: false,
      queryHash: '421A7F3B',
      planCacheShapeHash: '421A7F3B',
      planCacheKey: '0AB69667',
      optimizationTimeMillis: 1,
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      prunedSimilarIndexes: false,
      winningPlan: {
        isCached: false,
        stage: 'COLLSCAN',
        filter: {
          category: {
            '$eq': 'tablet'
          }
        },
        direction: 'forward'
```

```
> db.tempProducts.createIndex({ category: 1 })
< category_1
> db.tempProducts.find({ category: "tablet" }).explain("executionStats")
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'vit.tempProducts',
      parsedQuery: {
        category: {
          '$eq': 'tablet'
        }
      },
      indexFilterSet: false,
      queryHash: '421A7F3B',
      planCacheShapeHash: '421A7F3B',
      planCacheKey: 'E8986359',
      optimizationTimeMillis: 2,
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      prunedSimilarIndexes: false,
      winningPlan: {
        isCached: false,
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
```

```
> db.tempProducts.createIndex({ category: 1, price: -1 })
< category_1_price_-1
> db.tempProducts.find({ category: "tablet" }).sort({ price: -1 })
< {
    _id: ObjectId('6841a4d6c14e48208d9dcc22'),
    name: 'xTablet',
    category: 'tablet',
    price: 899,
    tags: [
      'electronics',
      'tablet'
    ],
    createdAt: 2023-01-10T00:00:00.000Z
  }
  {
    _id: ObjectId('6841a4d6c14e48208d9dcc23'),
    name: 'SmartTablet',
    category: 'tablet',
    price: 899,
    tags: [
      'smart',
      'tablet'
    ],
    createdAt: 2024-01-15T00:00:00.000Z
  }
  {
    _id: ObjectId('6841a4d6c14e48208d9dcc24'),
```

```java
public class MongoDb {

    public static void main(String[] args) {
        try {
            MongoClient db
                = new MongoClient( host: "localhost",  port: 27017);

            MongoCredential credential;
            credential
                = MongoCredential
                      .createCredential(
                          userName: "GFGUser",  database: "mongoDb",
                          "password".toCharArray());
            System.out.println(
                "Successfully Connected"
                + " to the database");

            MongoDatabase database
                = db.getDatabase( databaseName: "mongoDb");
            System.out.println("Credentials are: "
                               + credential);
        }
        catch (Exception e) {
            System.out.println(
                "Connection establishment failed");
            System.out.println(e);
        }
```

```
ngoDb ×
0: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWait
cessfully Connected to the database
dentials are: MongoCredential{mechanism=null, userName='GFGUser', source='mongoDb', password=<hidden>, mechanismProperties=<hidden>}
```