1-1

```
use mydatabase
```

```
db.current
```

```
db.dropDatabase()
```

1-2

```
db.customers.insert(
	{ "firstName":"John",
	  "lastName":"West",
	  "email":"john.west@mail.com",
	  "phone":"032345432134",
	  "BusinessType": ["Sell", "Sugar", "Drinks"],
	  "Reference":100,
	  "Company":"Coca-Cola"})
```

The following commands need to be issued from Terminal (Command Prompt), not the client window.

2-1

Windows

```
mongoimport --db mydatabase --jsonArray --collection transactions --drop --file transactions.json
```

MacOS

```
sudo ./mongoimport --db mydatabase --jsonArray --collection transactions --drop --file transactions.json
```

2-2

Windows

```
mongoimport --db mydatabase --jsonArray --collection transactions --mode=insert --file transactions.json
```

MacOS

```
sudo ./mongoimport --db mydatabase --jsonArray --collection transactions --mode=insert --file transactions.json
```

2-3

Windows

```
mongoimport --db mydatabase --jsonArray --collection transactions --mode=upsert --upsertFields=Id --file transaction_upsert.json
```

MacOS

```
sudo ./mongoimport --db mydatabase --jsonArray --collection transactions --mode=upsert --upsertFields=Id --file transaction_upsert.json
```

Check count of records after import

```
db.transactions.count()

db.transactions.find().pretty()
```

3

```
# 1. Find a record in transactions where name is tom

db.transactions.find({Name: 'Tom'})


# 2. Find a record in transactions where total payment amount is 400.
# When chaining key with dot, you need doublequote.

db.transactions.find({"Payment.Total": 400 })

db.transactions.find({"Payment.Total": {$eq: 400}})


# 3. Find a record in transactions collection where price is greater than 400

db.transactions.find({"Transaction.price": {$gt: 400} })

# can do equal or greater than, too.

db.transactions.find({"Transaction.price": {$gte: 400} })


# 4. Find a record in transacaions collection where note is null or missing

db.transactions.find({"Note": null})


# 5. Find a record where only Note key is missing

db.transactions.find({Note: {$exists: false } })


#6. Null only

db.transactions.find({Note: {$type: 10 } })
```

4

4-1. Inserting a record into transaction

```
db.transactions.insert(

   {

   "Id": 110,

   "Name": "Inserted Record",

   "TransactionId": "tranNew1",

  "Transaction": [

   {

   "ItemId":"c324",

   "price": 456

   },
   {
   "ItemId":"d456",
   "price": 543
   }
  ],
  "Subscriber": false,
  "Payment": {
   "Type": "Debit-Card",
   "Total": 999,
   "Success": true
  },
  "Note":'Hello World'
})
```

```
# check the record
db.transactions.find({Id:110})
```

4-2. Updating the newly inserted record above.

```
db.transactions.update({Id:110},{$set:{Name:'Updated Record',Note:'Updated!'}})
```

4-3. Deleting record

```
db.transactions.remove({Id:110})
```

5 Aggregation problems

5-1. Calculate the total transaction amount by aggregating Payment.Total in all records

```
db.transactions.aggregate({
   $group: {
      _id: '',
      TotalRevenue: { $sum: '$Payment.Total' }
   }
 })
```

5-2. Aggregate per record by aggregating Transaction.price

```
db.transactions.aggregate([
  {
    $project: {
      revenueTotal: { $sum: "$Transaction.price"},
    }
  }
])
```

5-3. Aggregate per payment type by adding up Payment.Total

```
db.transactions.aggregate([
    {
      $group:
       {
        _id: "$Payment.Type",
         totalAmount: { $sum: "$Payment.Total" },
         count: { $sum: 1 }
       }
    }
])
```

5-4. Find the max id

```
db.transactions.aggregate([
   {
      $group:
      {
          _id: '',
         maxId: {$max: "$Id"}
      }
   }
])
```

5-5. Find the max Transaction.price

# Transaction.price is an array. So find the array containing maximum price and then extract the max price from the array with another $max.

```
db.transactions.aggregate([
   {
      $group:
      {
         _id: '',
         maxPrice: {$max: {$max: "$Transaction.price"}}
      }
   }
])
```