**Exercise:** Demonstration of CRUD Operations on MongoDB

Q1. Perform the following tasks

1. Create a Database “Demo\_Database”
2. Create a Collection “Sample\_Collection” in the Demo\_Database and import the data from the given csv file.

# Taken Medical Expenses Dataset

3. Perform Insert, Read, Update and Remove operations using Sample\_Collection

# Query

db.Sample\_Collections.insertOne({age:35,sex:'female',bmi:37.5,children:1,smoker:'yes',regio n:'northwest',expenses:12450.12})

# Output

{

acknowledged: true, insertedId: ObjectId("65165e35f9fffcb7fdd24c6a")

}

# Viewing the inserted data

db.Sample\_Collections.find({expenses:12450.12})

[

{

\_id: ObjectId("6519cd77ee383e1c8e976704"), age: 35, sex: 'female', bmi: 37.5, children: 1, smoker: 'yes', region: 'northwest', expenses: 12450.12

}

]

# Read Operations

1) Finding all records for smokers with high expenses

**Query**

db.Sample\_Collections.find({smoker:'yes',expenses:{$gt:62000}})

## Output

[

{

\_id: ObjectId("6519cc62717f9b7098d69d30"), age: 54, sex: 'female', bmi: 47.4, children: 0, smoker: 'yes', region: 'southeast', expenses: 63770.43

},

{

\_id: ObjectId("6519cc62717f9b7098d6a025"), age: 45, sex: 'male', bmi: 30.4, children: 0, smoker: 'yes', region: 'southeast', expenses: 62592.87

}

]

2) Retrieving the average bmi for male and female individuals

**Query**

db.Sample\_Collections.aggregate([{$group:{\_id:'$sex',averageBMI:{$avg:'$bmi'}}}])

## Output

[

{ \_id: 'female', averageBMI: 30.390497737556565 },

{ \_id: 'male', averageBMI: 30.94526627218935 }

]

3) Finding the oldest person in the dataset

**Query**

db.Sample\_Collections.aggregate([{$group:{\_id:null,oldestPerson:{$max:'$age'}}}])

## Output

[ { \_id: null, oldestPerson: 64 } ]

4) List the regions with average expenses

**Query**

db.Sample\_Collections.aggregate([{$group:{\_id:'$region',highestexpenses:{$avg:'$expenses'}}}])

## Output

[

{ \_id: 'southwest', highestexpenses: 12346.937907692307 },

{ \_id: 'southeast', highestexpenses: 14735.411538461538 },

{ \_id: 'northwest', highestexpenses: 12417.675 },

{ \_id: 'northeast', highestexpenses: 13406.384691358024 }

]

5) Finding male and female individuals with highest expenses

**Query**

db.Sample\_Collections.aggregate([{$group:{\_id:'$sex',highestexpenses:{$max:'$expenses'}}}])

## Output

[

{ \_id: 'female', highestexpenses: 63770.43 },

{ \_id: 'male', highestexpenses: 62592.87 }

# Update Operations

1) Updating bmi and smoking status for individuals with high expenses

## Query

db.Sample\_Collections.updateMany({ expenses: { $gt: 12000 } }, { $set: { bmi: 26.0, smoker: "no" } })

## Output

{

acknowledged: true, insertedId: null, matchedCount: 493, modifiedCount: 493, upsertedCount: 0

}

2) Update region for individuals with higher BMI

**Query** db.Sample\_Collections.updateMany({bmi:36},{$set:{region:"central"}})

## Output

[

{

\_id: ObjectId("6519cc62717f9b7098d69b9c"), age: 22, sex: 'female', bmi: 36, children: 0, smoker: 'no', region: 'central', expenses: 2166.73

},

{

\_id: ObjectId("6519cc62717f9b7098d69dfd"), age: 47, sex: 'female', bmi: 36, children: 1, smoker: 'no', region: 'central', expenses: 8556.91

},

{

\_id: ObjectId("6519cc62717f9b7098d69e1f"), age: 51, sex: 'male', bmi: 36, children: 1, smoker: 'no', region: 'central', expenses: 9386.16

},

{

\_id: ObjectId("6519cc62717f9b7098d69e54"), age: 42, sex: 'male', bmi: 36, children: 2, smoker: 'no', region: 'central', expenses: 7160.33

}

]

3) Updating the number of children for families

**Query**

db.Sample\_Collections.updateMany({ children: { $gt: 0 } },{ $inc: { children: 1 } })

## Output

{ acknowledged: true, insertedId: null, matchedCount: 765, modifiedCount: 765, upsertedCount: 0

}

4) Updating records with missing data

**Query**

db.Sample\_Collections.updateMany({region:{$exists:false}},{$set:{region:'Unknown'}})

## Output

{

acknowledged: true, insertedId: null, matchedCount: 0, modifiedCount: 0, upsertedCount: 0

}

**Findings:** Since there is no missing data in region feature, updated count is 0.

5) Flagging individuals with high expenses

**Query**

db.Sample\_Collections.updateMany({expenses:{$gt:50000}},{$set:{highexpenses:true}})

## Output

{

acknowledged: true, insertedId: null, matchedCount: 7, modifiedCount: 7, upsertedCount: 0

}

## Proof

[

{

\_id: ObjectId("6519cc62717f9b7098d69b33"), age: 28, sex: 'male', bmi: 26, children: 2, smoker: 'no', region: 'southwest', expenses: 51194.56, highexpenses: true

},

]

# Delete Operations

1) Delete records of all individuals having no children and with low expenses

**Query** db.Sample\_Collections.deleteMany({children:0},{expenses:{$lt:3000}})

## Output

{ acknowledged: true, deletedCount: 574 }

2) Removing records of non-smokers with low expenses

**Query**

db.Sample\_Collections.deleteMany({smoker:'no'},{expenses:{$lt:40000}})

## Output

{ acknowledged: true, deletedCount: 765 }

3) Removing the records of smokers with high expenses

**Query**

db.Sample\_Collections.deleteMany({smoker:'yes'},{expenses:{$gt:52000}})

## Output

{ acknowledged: true, deletedCount: 0 }

**Findings:** There is no such kind of records present in the dataset, so the deleted count is 0. 4) Deleting records of individuals with high expenses

**Query**

db.Sample\_Collections.deleteMany({expenses:{$gt:62000}})

## Output

{ acknowledged: true, deletedCount: 0 }

5) Deleting all the records from the collection

**Query**

db.Sample\_Collections.deleteMany({})

## Output

db.Sample\_Collections.find()

Since we removed all the records from the collections nothing will be displayed.