**Databse practical sol**

**MongoDB**

**Slip1**

**Q2:-** **Model the following Property system as a document database. Consider a set of Property, Owner. One owner can buy many properties.**

**Sol:-**

Use property\_management

db.createCollection("property")

db.property.insertMany([

{ "\_id": 1, "area": "Mumbai", "rate": 150000, "owner\_id": 1 },

{ "\_id": 2, "area": "Pune", "rate": 90000, "owner\_id": 2 },

{ "\_id": 3, "area": "Nashik", "rate": 120000, "owner\_id": 3 },

{ "\_id": 4, "area": "Nagpur", "rate": 80000, "owner\_id": 4 },

{ "\_id": 5, "area": "Mumbai", "rate": 200000, "owner\_id": 2 }

])

db.createCollection("owner")db.owner.insertMany([

{ "\_id": 1, "name": "Mr. Singh" },

{ "\_id": 2, "name": "Mr. Patil" },

{ "\_id": 3, "name": "Ms. Rao" },

{ "\_id": 4, "name": "Mr. Kumar" },

{ "\_id": 5, "name": "Mrs. Sharma" }

])

**a)Display area wise property details. [3]**

db.property.aggregate([

{ $group: { \_id: "$area", properties: { $push: "$$ROOT" } } }

])

**b)Display property owned by 'Mr.Patil' having minimum rate [3]**

db.property.find({ owner\_id: db.owner.findOne({ name: "Mr. Patil" }).\_id }).sort({ rate: 1 }).limit(1)

**c)Give the details of owner whose property is at “Nashik”. [4]**

db.owner.findOne({ \_id: db.property.findOne({ area: "Nashik" }).owner\_id })

**d)Display area of property whose rate is less than 100000. [4]**

db.owner.findOne({ \_id: db.property.findOne({ area: "Nashik" }).owner\_id })

**slip2**

**Q2 Model the following system as a document database. Consider a database of newspaper, publisher, and city. Different publisher publishes various newspapers in different cities**

**Sol:-**

use newspaper\_management

db.createCollection("newspapers")

db.createCollection("publishers")

db.createCollection("cities")

db.newspapers.insertMany([

{ "\_id": 1, "name": "The Times", "language": "English", "city": "Mumbai", "sale": 5000 },

{ "\_id": 2, "name": "Maharashtra Times", "language": "Marathi", "city": "Pune", "sale": 7000 },

{ "\_id": 3, "name": "Gujarat Samachar", "language": "Gujarati", "city": "Ahmedabad", "sale": 6000 },

{ "\_id": 4, "name": "Nashik Times", "language": "Marathi", "city": "Nashik", "sale": 4500 },

{ "\_id": 5, "name": "Pune Mirror", "language": "English", "city": "Pune", "sale": 5500 }

])

**a. List all newspapers available “NASHIK” city [3]**

db.newspapers.find({ city: "Nashik" })

**b. List all the newspaper of “Marathi” language [3]**

db.newspapers.find({ city: "Nashik" })

**c. Count no. of publishers of “Gujrat” state [4]**

db.publishers.count({ state: "Gujarat" })

**d. Write a cursor to show newspapers with highest sale in Maharashtra state[4]**

var cursor = db.newspapers.find({}).sort({ sale: -1 }).limit(1);

while(cursor.hasNext()) {

printjson(cursor.next());

**Slip 3**

**Q2 Model the following system as a document database. Consider employee and department’s information.**

**Sol:-**

use company\_management

// Create employees collection

db.createCollection("employees")

db.employees.insertMany([

{ "\_id": 1, "name": "John Doe", "department\_id": 1, "salary": 60000 },

{ "\_id": 2, "name": "Jane Smith", "department\_id": 1, "salary": 55000 },

{ "\_id": 3, "name": "Alice Johnson", "department\_id": 2, "salary": 62000 },

{ "\_id": 4, "name": "Bob Williams", "department\_id": 2, "salary": 58000 },

{ "\_id": 5, "name": "Emily Brown", "department\_id": 3, "salary": 63000 }

])

// Create departments collection

db.createCollection("departments")

db.departments.insertMany([

{ "\_id": 1, "name": "Sales", "employees": [1, 2] },

{ "\_id": 2, "name": "Marketing", "employees": [3, 4] },

{ "\_id": 3, "name": "Finance", "employees": [5] }

])

**a. Display name of employee who has highest salary [3]**

db.employees.find().sort({ salary: -1 }).limit(1).pretty()

**b. Display biggest department with max. no. of employees [3]**

db.departments.aggregate([

{ $project: { name: 1, numEmployees: { $size: "$employees" } } },

{ $sort: { numEmployees: -1 } }, { $limit: 1 }])

**c. Write a cursor which shows department wise employee information [4]**

var cursor = db.departments.find();

while(cursor.hasNext()) {

var department = cursor.next();

print("Department: " + department.name);

db.employees.find({ department\_id: department.\_id }).forEach(printjson);

}

**d. List all the employees who work in Sales dept and salary > 50000 [4]**

db.employees.find({ department\_id: 1, salary: { $gt: 50000 } }).pretty()

**slip4**

**Q2 Model the following information system as a document database. Consider hospitals around Nashik. Each hospital may have one or more specializations like Pediatric, Gynaec, Orthopedic, etc. A person can recommend/provide review for a hospital. A doctor can give service to one or more hospitals.**

**Sol:-**

use hospital\_management

// Create hospitals collection

db.createCollection("hospitals")

db.hospitals.insertMany([

{ name: "City Hospital", city: "Nashik", specializations: ["Pediatric", "Orthopedic"], rating: 4.5 },

{ name: "ABC Clinic", city: "Nashik", specializations: ["Gynaec", "Orthopedic"], rating: 4.0 },

{ name: "XYZ Hospital", city: "Mumbai", specializations: ["Cardiology", "Neurology"], rating: 4.8 },

// Add more hospital documents as needed

])

// Create specializations collection

db.createCollection("specializations")

db.specializations.insertMany([

{ name: "Pediatric" },

{ name: "Orthopedic" },

{ name: "Gynaec" },

{ name: "Cardiology" },

{ name: "Neurology" },

// Add more specialization documents as needed ])

// Create reviews collection

db.createCollection("reviews")

db.reviews.insertMany([

{ hospital\_id: 1, reviewer\_name: "John Doe", rating: 5 },

{ hospital\_id: 2, reviewer\_name: "Jane Smith", rating: 4 },])

// Create doctors collection

db.createCollection("doctors")

db.doctors.insertMany([

{ name: "Dr. John Doe", hospitals: [1, 2] },

{ name: "Dr. Deshmukh", hospitals: [2, 3] },

// Add more doctor documents as needed

])

1. **List the names of hospitals with………… specialization. [3]**

db.hospitals.find({ specializations: "Pediatric" }, { name: 1, \_id: 0 })

1. **List the Names of all hospital located in ……. city [3]**

db.hospitals.find({ city: "Nashik" }, { name: 1, \_id: 0 })

1. **List the names of hospitals where Dr. Deshmukh visits [4]**

var doctor = db.doctors.findOne({ name: "Dr. Deshmukh" });

db.hospitals.find({ \_id: { $in: doctor.hospitals } }, { name: 1, \_id: 0 })

1. **List the names of hospitals whose rating >=4 [4]**

db.hospitals.find({ rating: { $gte: 4 } }, { name: 1, \_id: 0 })

**slip5**

**Q2 Model the following database. Many employees working on one project. A company has various ongoing projects**.

**Sol:-**

use company\_management

db.createCollection("employees")

// Insert documents into the "employees" collection

db.employees.insertMany([

{ "\_id": 1, "name": "John Doe", "project\_id": 1 },

{ "\_id": 2, "name": "Jane Smith", "project\_id": 2 },

// Add more employee documents as needed])

db.createCollection("projects")

// Insert documents into the "projects" collection

db.projects.insertMany([

{ "\_id": 1, "name": "Project A", "project\_type": "Type X", "duration": 4 },

{ "\_id": 2, "name": "Project B", "project\_type": "Type Y", "duration": 5 },

// Add more project documents as needed ])

**a. List all names of projects where Project\_type =….. [3]**

db.projects.find({ project\_type: "Type X" }, { name: 1, \_id: 0 })

**b. List all the projects with duration greater than 3 months [3]**

db.projects.find({ duration: { $gt: 3 } })

**c. Count no. of employees working on ……..project [4]**

db.employees.count({ project\_id: 1 }) // Count employees working on Project A

**d.List the names of projects on which Mr. Patil is working [4]**

var patilProjectIds = db.employees.find({ name: "Mr. Patil" }).map(e => e.project\_id);

db.projects.find({ \_id: { $in: patilProjectIds } }, { name: 1, \_id: 0 })

**slip 6**

**Q2 Model the following information as a document database. A customer can take different policies and get the benefit. There are different types of policies provided by various companies**

**Sol:-**

use insurance\_company\_management // Switch to the desired database (replace "insurance\_company\_management" with your database name)

// Create the "customers" collection

db.createCollection("customers")

db.customers.insertMany([

{ "\_id": 1, "name": " Komal Jeevan ", "policy\_id": 1, "premium\_amount": 5000 },

{ "\_id": 2, "name": "Jane Smith", "policy\_id": 2, "premium\_amount": 6000 },

// Add more customer documents as needed

])

// Create the "policies" collection

db.createCollection("policies")

db.policies.insertMany([

{ "\_id": 1, "type": "Monthly", "name": "Komal Jeevan", "company\_id": 1, "benefit": "Life insurance" },

{ "\_id": 2, "type": "Half Yearly", "name": "Secure Life", "company\_id": 2, "benefit": "Health insurance" },

// Add more policy documents as needed

])

// Create the "companies" collection

db.createCollection("companies")

db.companies.insertMany([

{ "\_id": 1, "name": "ABC Insurance", "type": "Insurance" },

{ "\_id": 2, "name": "XYZ Financial Services", "type": "Financial Services" },

// Add more company documents as needed

])

1. **List the details of customers who have taken “Komal Jeevan” Policy**

db.customers.find({ "policy\_id": db.policies.findOne

({ "name": "Komal Jeevan" }).\_id })

1. **Display average premium amount**

db.customers.aggregate([

{ $group: { \_id: null, average\_premium: { $avg: "$premium\_amount" } } }

])

1. **Increase the premium amount by 5% for policy type=”Monthly”**

db.customers.updateMany(

{ "policy\_id": { $in: db.policies.find({ "type": "Monthly" }).map(p => p.\_id) } },

{ $mul: { "premium\_amount": 1.05 } }

)

1. **Count no. of customers who have taken policy type “half yearly”.**

db.customers.count({ "policy\_id": { $in: db.policies.find({ "type": "Half Yearly" }).map(p => p.\_id) } })

**slip7**

**Q2 . Model the following information as a document database. A customer operates his bank account, does various transactions and get the banking services**

**Sol:-**

use banking\_system

// Create the "customers" collection

db.createCollection("customers")

db.customers.insertMany([

{ "\_id": 1, "first\_name": "Suraj", "last\_name": "Bhosale", },

{ "\_id": 2, "first\_name": "Jane", "last\_name": "Smith", },

// Add more customer documents as needed

])

// Create the "accounts" collection

db.createCollection("accounts")

db.accounts.insertMany([

{ "\_id": 1, "customer\_id": 1, "account\_type": "Savings", "balance": 5000, },

{ "\_id": 2, "customer\_id": 2, "account\_type": "Checking", "balance": 6000,},

// Add more account documents as needed

])

// Create the "transactions" collection

db.createCollection("transactions")

db.transactions.insertMany([

{ "\_id": 1, "account\_id": 1, "type": "Deposit", "amount": 1000, "date": ISODate("2020-01-01"), },

{ "\_id": 2, "account\_id": 2, "type": "Withdrawal", "amount": 500, "date": ISODate("2022-01-02"), },

// Add more transaction documents as needed

])

1. **List names of all customers whose first name starts with a “S”**

db.customers.find({ "first\_name": /^S/i }, { "first\_name": 1, "last\_name": 1, "\_id": 0 })

1. **List all customers who has open an account on 1/1/2020 in \_\_\_branch**

db.customers.aggregate([

{ $lookup: { from: "accounts", localField: "\_id", foreignField: "customer\_id", as: "customer\_accounts" } },

{ $unwind: "$customer\_accounts" },

{ $match: { "customer\_accounts.opening\_date": ISODate("2020-01-01"), "customer\_accounts.branch": "BranchName" } },

{ $project: { "first\_name": 1, "last\_name": 1, "\_id": 0 } }

])

1. **List the names customers where acctype=”Saving”**

db.customers.aggregate([

{ $lookup: { from: "accounts", localField: "\_id", foreignField: "customer\_id", as: "customer\_accounts" } },

{ $unwind: "$customer\_accounts" },

{ $match: { "customer\_accounts.account\_type": "Savings" } },

{ $project: { "first\_name": 1, "last\_name": 1, "\_id": 0 } }

])

**d. Count total no. of loan account holder of …….branch [4] 20Viva 5**

db.accounts.count({ "branch": "BranchName", "account\_type": "Loan" })

**slip8**

**Q2 Model the following inventory information as a document database. The inventory keeps track of various items. The items are tagged in various categories. Items may be kept in various warehouses and each warehouse keeps track of the quantity of the item.**

**Sol:-**

use inventory\_system // Switch to the desired database (replace "inventory\_system" with your database name)

// Create the "items" collection

db.createCollection("items")

db.items.insertMany([

{ "\_id": 1, "name": "Widget", "tags": ["electronics", "gadget"], "status": "A", "height": 10, },

{ "\_id": 2, "name": "Planner", "tags": ["office", "stationery"], "status": "B", "height": 8, },

// Add more item documents as needed

])

// Create the "categories" collection

db.createCollection("categories")

db.categories.insertMany([

{ "\_id": 1, "name": "Electronics", },

{ "\_id": 2, "name": "Office Supplies", },

// Add more category documents as needed

])

// Create the "warehouses" collection

db.createCollection("warehouses")

db.warehouses.insertMany([

{ "\_id": 1, "name": "Main Warehouse", "location": "City A", },

{ "\_id": 2, "name": "Secondary Warehouse", "location": "City B", },

// Add more warehouse documents as needed

])

// Create the "inventory" collection

db.createCollection("inventory")

db.inventory.insertMany([

{ "\_id": 1, "item\_id": 1, "warehouse\_id": 1, "quantity": 500, },

{ "\_id": 2, "item\_id": 2, "warehouse\_id": 2, "quantity": 15, },

// Add more inventory documents as needed

])

1. **List all the items qty is greater than 300**

db.inventory.aggregate([

{ $lookup: { from: "items", localField: "item\_id", foreignField: "\_id", as: "item\_info" } },

{ $match: { "quantity": { $gt: 300 } } },

{ $unwind: "$item\_info" },

{ $project: { "item\_info.name": 1, "quantity": 1, "\_id": 0 } }])

1. **List all items which have tags less than 5**

db.items.find({ "tags": { $exists: true, $size: { $lt: 5 } } })

1. **List all items having status equal to “B” or having quantity less than 50 and height of the product should be greater than 8**

db.inventory.aggregate([

{ $lookup: { from: "items", localField: "item\_id", foreignField: "\_id", as: "item\_info" } },

{ $match: { $or: [ { "item\_info.status": "B" }, { "quantity": { $lt: 50 }, "item\_info.height": { $gt: 8 } } ] } },

{ $unwind: "$item\_info" },

{ $project: { "item\_info.name": 1, "quantity": 1, "\_id": 0 } }

])

**D. Find all warehouse that keeps item “Planner” and having in stock quantity less than 20**

db.inventory.aggregate([

{ $match: { "item\_id": { $eq: db.items.findOne({ "name": "Planner" }).\_id }, "quantity": { $lt: 20 } } },

{ $lookup: { from: "warehouses", localField: "warehouse\_id", foreignField: "\_id", as: "warehouse\_info" } },

{ $unwind: "$warehouse\_info" },

{ $project: { "warehouse\_info.name": 1, "quantity": 1, "\_id": 0 } }

])