1.For the given Library database

BOOK (Book\_ISBN [PK], Title[Not Null], Publisher\_ Name, price[Check Price&gt;0],

Date\_Of\_Publication,Book\_Copy ),

BOOK\_AUTHORS (Book\_ISBN [PK,FK]Author\_Name [PK], Author\_City)

Solve the following

a) Create view BOOK\_AUTHOR\_INFO consisting Book\_ISBN, Title from

BOOK Table and Author\_Name from BOOK\_AUTHORS Table in ascending

order of ISBN no.

b) Create an index on Book\_Author on table on attribute “Author\_Name”.

c) Create table Book\_Auto\_Increment (BookID int Auto\_increament=100, Book

Name) insert five records in table.

d) Delete the book from Book table written by Author ‘Korth’.

e) Select Book Names from table Book whose copies are in between 10 to 15.

ANS:

**a) Create a View BOOK\_AUTHOR\_INFO**

Here's the MySQL-compatible statement:

sql

Copy code

CREATE VIEW BOOK\_AUTHOR\_INFO AS

SELECT b.Book\_ISBN, b.Title, a.Author\_Name

FROM BOOK b

JOIN BOOK\_AUTHORS a ON b.Book\_ISBN = a.Book\_ISBN

ORDER BY b.Book\_ISBN ASC;

**b) Create an Index on Author\_Name**

This statement is also compatible with MySQL:

sql

Copy code

CREATE INDEX idx\_author\_name ON BOOK\_AUTHORS (Author\_Name);

**c) Create a Table with Auto-Increment**

To create the table with an auto-increment starting at 100 and insert five records, use:

sql

Copy code

CREATE TABLE Book\_Auto\_Increment (

BookID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

BookName VARCHAR(255) NOT NULL

) AUTO\_INCREMENT=100;

INSERT INTO Book\_Auto\_Increment (BookName) VALUES

('Book 1'),

('Book 2'),

('Book 3'),

('Book 4'),

('Book 5');

**d) Delete Books by Author 'Korth'**

To delete books from the BOOK table authored by 'Korth':

sql

Copy code

DELETE FROM BOOK

WHERE Book\_ISBN IN (

SELECT Book\_ISBN FROM BOOK\_AUTHORS WHERE Author\_Name = 'Korth'

);

**e) Select Book Names with Copies Between 10 and 15**

To select book titles where the number of copies is between 10 and 15:

sql

Copy code

SELECT Title

FROM BOOK

WHERE Book\_Copy BETWEEN 10 AND 15;

2. For the given Library database BOOK (Book\_ISBN [PK], Title[Not Null], Publisher\_ Name, price[Check Price&gt;0], Date\_Of\_Publication,Book\_Copy ), BOOK\_AUTHORS (Book\_ISBN [PK,FK]Author\_Name [PK], Author\_City) Solve the following : a) Select Book\_ISBN, Title, Author\_Name from relations Book and Book\_Authors INNER JOIN on attribute Book\_ISBN. b) Select Book\_ISBN, Title, Publisher, Author\_Name from relations Book and Book\_Authors LEFT OUTER JOIN on attribute Book\_ISBN. c) Select Book\_ISBN, Title, Publisher, Author\_Name from relations Book and Book\_Authors RIGHT OUTER JOIN on attribute Book\_ISBN. d) Select Book\_ISBN, Title from relation Book whose author is living in City =’Pune’. e) Select Book\_ISBN, Title from relation Book, which written by more than 2 Authors.

**a) Select Book\_ISBN, Title, Author\_Name using INNER JOIN**

This query selects the Book\_ISBN, Title, and Author\_Name from both tables, only including books that have authors.

sql

Copy code

SELECT b.Book\_ISBN, b.Title, a.Author\_Name

FROM BOOK b

INNER JOIN BOOK\_AUTHORS a ON b.Book\_ISBN = a.Book\_ISBN;

**b) Select Book\_ISBN, Title, Publisher\_Name, Author\_Name using LEFT OUTER JOIN**

This query selects all books, including those that may not have any associated authors.

sql

Copy code

SELECT b.Book\_ISBN, b.Title, b.Publisher\_Name, a.Author\_Name

FROM BOOK b

LEFT OUTER JOIN BOOK\_AUTHORS a ON b.Book\_ISBN = a.Book\_ISBN;

**c) Select Book\_ISBN, Title, Publisher\_Name, Author\_Name using RIGHT OUTER JOIN**

This query includes all authors and their associated books, even if some authors do not have any corresponding books.

sql

Copy code

SELECT b.Book\_ISBN, b.Title, b.Publisher\_Name, a.Author\_Name

FROM BOOK b

RIGHT OUTER JOIN BOOK\_AUTHORS a ON b.Book\_ISBN = a.Book\_ISBN;

**d) Select Book\_ISBN, Title where the author lives in City = 'Pune'**

This query retrieves the ISBN and title of books written by authors who reside in Pune.

sql

Copy code

SELECT b.Book\_ISBN, b.Title

FROM BOOK b

INNER JOIN BOOK\_AUTHORS a ON b.Book\_ISBN = a.Book\_ISBN

WHERE a.Author\_City = 'Pune';

**e) Select Book\_ISBN, Title for books written by more than 2 authors**

This query retrieves the ISBN and title of books that have more than two authors associated with them.

sql

Copy code

SELECT b.Book\_ISBN, b.Title

FROM BOOK b

INNER JOIN BOOK\_AUTHORS a ON b.Book\_ISBN = a.Book\_ISBN

GROUP BY b.Book\_ISBN, b.Title

HAVING COUNT(a.Author\_Name) > 2;

3. For the given Library database BOOK (Book\_ISBN [PK], Title[Not Null], Publisher\_ Name, price[Check Price&gt;0], Date\_Of\_Publication,Book\_Copy ), BOOK\_AUTHORS (Book\_ISBN [PK,FK]Author\_Name [PK], Author\_City) Solve the following i) Display name of publishers as per no of books published by them in ascending order. ii) Get publisher names who published at least one book written by author name like ‘K%’. iii) Get book name and Authors names where book written by maximum authors. iv) Get publisher names accordingly books published alphabetically v) Find the no of books published in 01 Jan 2014 to till date. using mysql

**ChatGPT said:**

ChatGPT

Here are the SQL queries for your requests based on the BOOK and BOOK\_AUTHORS tables in your Library database:

**i) Display names of publishers based on the number of books published in ascending order**

This query counts the number of books for each publisher and sorts the results in ascending order:

sql

Copy code

SELECT Publisher\_Name, COUNT(\*) AS Number\_of\_Books

FROM BOOK

GROUP BY Publisher\_Name

ORDER BY Number\_of\_Books ASC;

**ii) Get publisher names who published at least one book written by an author whose name starts with 'K'**

This query retrieves publisher names based on books written by authors with names that start with 'K':

sql

Copy code

SELECT DISTINCT b.Publisher\_Name

FROM BOOK b

JOIN BOOK\_AUTHORS a ON b.Book\_ISBN = a.Book\_ISBN

WHERE a.Author\_Name LIKE 'K%';

**iii) Get book name and authors' names where the book is written by the maximum number of authors**

This query identifies the book(s) with the highest number of associated authors:

sql

Copy code

SELECT b.Title, GROUP\_CONCAT(a.Author\_Name SEPARATOR ', ') AS Authors

FROM BOOK b

JOIN BOOK\_AUTHORS a ON b.Book\_ISBN = a.Book\_ISBN

GROUP BY b.Book\_ISBN, b.Title

HAVING COUNT(a.Author\_Name) = (

SELECT MAX(AuthorCount)

FROM (

SELECT COUNT(\*) AS AuthorCount

FROM BOOK\_AUTHORS

GROUP BY Book\_ISBN

) AS AuthorCounts

);

**iv) Get publisher names along with books published, sorted alphabetically**

This query retrieves all publisher names and associated books, sorted by publisher name:

sql

Copy code

SELECT b.Publisher\_Name, b.Title

FROM BOOK b

ORDER BY b.Publisher\_Name, b.Title;

**v) Find the number of books published from January 1, 2014, to the current date**

This query counts the number of books published within the specified date range:

sql

Copy code

SELECT COUNT(\*) AS Number\_of\_Books

FROM BOOK

WHERE Date\_Of\_Publication >= '2014-01-01';

4. Consider insurance database with following schema : person(driver-id, name, address) car(license, model, year) accident (report - no, date, location) owns(driver-id,license) participated(driver-id,car,report-no,damage-amount) Write a query in SQL for following requirements : i) Find the total no. of people who owned cars that were involved in accidents in 2016. ii) Retrieve the name of person whose address contains Pune. iii) Find the name of persons having more than two cars. in mysql

**ChatGPT said:**

ChatGPT

Here are the SQL queries to meet the requirements for the insurance database you described:

**i) Find the total number of people who owned cars that were involved in accidents in 2016**

This query will count distinct drivers who owned cars involved in accidents during the year 2016:

sql

Copy code

SELECT COUNT(DISTINCT o.driver-id) AS Total\_People

FROM owns o

JOIN participated p ON o.driver-id = p.driver-id

JOIN accident a ON p.report-no = a.report-no

WHERE YEAR(a.date) = 2016;

**ii) Retrieve the name of the person whose address contains 'Pune'**

This query selects the names of individuals whose address includes the substring 'Pune':

sql

Copy code

SELECT name

FROM person

WHERE address LIKE '%Pune%';

**iii) Find the names of persons having more than two cars**

This query retrieves names of persons who own more than two cars by grouping and filtering:

sql

Copy code

SELECT p.name

FROM person p

JOIN owns o ON p.driver-id = o.driver-id

GROUP BY p.driver-id, p.name

HAVING COUNT(o.license) > 2;

5. For the given Employee database EmployeeInfo(EmpID[PK],EmpFname,EmpLname,Department,Project,Address,DOB,Ge nder) EmployeePosition(EmpID[FK],EmpPosition,DateOfJoining,Salary) i. Write a query to fetch the EmpFname from the EmployeeInfo table in the upper case and use the ALIAS name as EmpName. ii. Write a query to fetch the number of employees working in the department ‘HR’. iii. Write q query to find all the employees whose salary is between 50000 to 100000 iv. Write a query to find the names of employees that begin with ‘S’ v. Write a query to fetch top N records.

**ChatGPT said:**

ChatGPT

Here are the SQL queries for the specified requirements based on the EmployeeInfo and EmployeePosition tables:

**i) Fetch EmpFname in upper case with an alias EmpName**

sql

Copy code

SELECT UPPER(EmpFname) AS EmpName

FROM EmployeeInfo;

**ii) Fetch the number of employees working in the department ‘HR’**

sql

Copy code

SELECT COUNT(\*) AS NumberOfEmployees

FROM EmployeeInfo

WHERE Department = 'HR';

**iii) Find all the employees whose salary is between 50,000 and 100,000**

sql

Copy code

SELECT ei.EmpFname, ei.EmpLname, ep.Salary

FROM EmployeeInfo ei

JOIN EmployeePosition ep ON ei.EmpID = ep.EmpID

WHERE ep.Salary BETWEEN 50000 AND 100000;

**iv) Find the names of employees that begin with ‘S’**

sql

Copy code

SELECT EmpFname, EmpLname

FROM EmployeeInfo

WHERE EmpFname LIKE 'S%';

**v) Fetch top N records**

To fetch the top N records, you can use the LIMIT clause. Replace N with the desired number of records. For example, to get the top 5 employees:

sql

Copy code

SELECT \*

FROM EmployeeInfo

LIMIT 5;

13. i) Retrieve all records from the Student table ii)Retrieve all records from the Subject table (iii) Find the number of students who have enrolled for the subject &quot;DBMS&quot;. (iv) Find all those faculty members who have not offered any subject. v) Find all subjects enrolled by Roll-no 1101.

**ChatGPT said:**

ChatGPT

Here are the SQL statements to create the specified tables and the queries to perform the required operations on the Student and Subject tables:

**Step 1: Create the Tables**

sql

Copy code

CREATE TABLE Student (

roll\_no INT PRIMARY KEY,

name VARCHAR(100),

subject\_name VARCHAR(100),

subject\_opted BOOLEAN

);

CREATE TABLE Subject (

faculty\_code INT PRIMARY KEY,

faculty\_name VARCHAR(100),

specialization VARCHAR(100)

);

**Step 2: Generate the Queries**

**i) Retrieve all records from the Student table**

sql

Copy code

SELECT \* FROM Student;

**ii) Retrieve all records from the Subject table**

sql

Copy code

SELECT \* FROM Subject;

**iii) Find the number of students who have enrolled for the subject "DBMS"**

sql

Copy code

SELECT COUNT(\*) AS NumberOfStudents

FROM Student

WHERE subject\_name = 'DBMS';

**iv) Find all faculty members who have not offered any subject**

Assuming that a faculty member is linked to subjects through the Subject table, you can find faculty members who do not have any entries in the Subject table by using a LEFT JOIN:

sql

Copy code

SELECT s.faculty\_code, s.faculty\_name

FROM Subject s

LEFT JOIN Student st ON s.faculty\_code = st.roll\_no

WHERE st.roll\_no IS NULL;

Note: If you need a different method to link faculty to subjects, please clarify.

**v) Find all subjects enrolled by Roll-no 1101**

sql

Copy code

SELECT subject\_name

FROM Student

WHERE roll\_no = 1101;

14. Create the following table : Item (item-code, item-name, qty-in-stock, reorder-level) Supplier (supplier-code, supplier-name, address) Can-supply(supplier-code, item-code) Generate queries to do the following : (i) Retrieve all records from the Item table ii)Retrieve all records from the Supplier table iii) Display all Items supplied by all suppliers. iv) Retrieve items where the quantity in stock is below the reorder level v)List all those suppliers who can supply the given item. vi) List all those items which cannot be supplied by given company

**ChatGPT said:**

ChatGPT

Here are the SQL statements to create the specified tables (Item, Supplier, and Can-supply) and the queries to perform the required operations.

**Step 1: Create the Tables**

sql

Copy code

CREATE TABLE Item (

item\_code INT PRIMARY KEY,

item\_name VARCHAR(100),

qty\_in\_stock INT,

reorder\_level INT

);

CREATE TABLE Supplier (

supplier\_code INT PRIMARY KEY,

supplier\_name VARCHAR(100),

address VARCHAR(255)

);

CREATE TABLE Can\_supply (

supplier\_code INT,

item\_code INT,

PRIMARY KEY (supplier\_code, item\_code),

FOREIGN KEY (supplier\_code) REFERENCES Supplier(supplier\_code),

FOREIGN KEY (item\_code) REFERENCES Item(item\_code)

);

**Step 2: Generate the Queries**

**i) Retrieve all records from the Item table**

sql

Copy code

SELECT \* FROM Item;

**ii) Retrieve all records from the Supplier table**

sql

Copy code

SELECT \* FROM Supplier;

**iii) Display all items supplied by all suppliers**

This query uses a JOIN to list all items along with their corresponding suppliers.

sql

Copy code

SELECT i.item\_name, s.supplier\_name

FROM Item i

JOIN Can\_supply cs ON i.item\_code = cs.item\_code

JOIN Supplier s ON cs.supplier\_code = s.supplier\_code;

**iv) Retrieve items where the quantity in stock is below the reorder level**

sql

Copy code

SELECT item\_name, qty\_in\_stock, reorder\_level

FROM Item

WHERE qty\_in\_stock < reorder\_level;

**v) List all those suppliers who can supply a given item**

Assuming you want to find suppliers for a specific item code, replace given\_item\_code with the actual item code:

sql

Copy code

SELECT s.supplier\_name

FROM Supplier s

JOIN Can\_supply cs ON s.supplier\_code = cs.supplier\_code

WHERE cs.item\_code = given\_item\_code;

**vi) List all those items which cannot be supplied by a given supplier**

Assuming you want to find items that cannot be supplied by a specific supplier code, replace given\_supplier\_code with the actual supplier code:

sql

Copy code

SELECT i.item\_name

FROM Item i

WHERE i.item\_code NOT IN (

SELECT cs.item\_code

FROM Can\_supply cs

WHERE cs.supplier\_code = given\_supplier\_code

);

MONGO

16. Create a collection named Book. (book\_isbn,title,punlisher\_name,author(Name, Address, Phone No[landline, mobile]), publisher\_city, price,copies) i. a. Add 5 documents in the collection with keys b. Give details of Books whose Publisher lives in “Pune”. c. Delete name Book from Book whose name start with “D” d. Change the city of publisher “Pearson” to “Pune”. e. Find the details of publisher named “Pearson”. mongodb

**ChatGPT said:**

ChatGPT

Here's how you can create a MongoDB collection named Book and perform the specified operations:

**Step 1: Create the Collection and Add Documents**

You can use the following commands in the MongoDB shell or a MongoDB client like MongoDB Compass.

javascript

Copy code

// Connect to the database (if needed)

use yourDatabaseName; // Replace with your actual database name

// Create the Book collection and insert 5 documents

db.Book.insertMany([

{

book\_isbn: "978-0136006633",

title: "Introduction to Algorithms",

publisher\_name: "MIT Press",

author: {

name: "Thomas H. Cormen",

address: "Cambridge, MA",

phone: {

landline: "123-456-7890",

mobile: "987-654-3210"

}

},

publisher\_city: "Cambridge",

price: 59.99,

copies: 10

},

{

book\_isbn: "978-0131101630",

title: "The C Programming Language",

publisher\_name: "Prentice Hall",

author: {

name: "Brian W. Kernighan",

address: "Pune, India",

phone: {

landline: "012-345-6789",

mobile: "987-654-3211"

}

},

publisher\_city: "Pune",

price: 45.00,

copies: 5

},

{

book\_isbn: "978-0201633610",

title: "Design Patterns",

publisher\_name: "Addison-Wesley",

author: {

name: "Erich Gamma",

address: "Boston, MA",

phone: {

landline: "234-567-8901",

mobile: "876-543-2109"

}

},

publisher\_city: "Boston",

price: 40.00,

copies: 8

},

{

book\_isbn: "978-0132350884",

title: "Clean Code",

publisher\_name: "Prentice Hall",

author: {

name: "Robert C. Martin",

address: "Chicago, IL",

phone: {

landline: "345-678-9012",

mobile: "765-432-1098"

}

},

publisher\_city: "Chicago",

price: 50.00,

copies: 7

},

{

book\_isbn: "978-0134685991",

title: "Effective Java",

publisher\_name: "Addison-Wesley",

author: {

name: "Joshua Bloch",

address: "Pune, India",

phone: {

landline: "456-789-0123",

mobile: "654-321-0987"

}

},

publisher\_city: "Pune",

price: 55.00,

copies: 12

}

]);

**Step 2: Query for Books Whose Publisher Lives in “Pune”**

javascript

Copy code

db.Book.find({ "author.address": /Pune/i });

**Step 3: Delete Books Whose Name Starts with “D”**

javascript

Copy code

db.Book.deleteMany({ title: { $regex: /^D/i } });

**Step 4: Change the City of Publisher “Pearson” to “Pune”**

javascript

Copy code

db.Book.updateMany(

{ publisher\_name: "Pearson" },

{ $set: { publisher\_city: "Pune" } }

);

**Step 5: Find the Details of Publisher Named “Pearson”**

javascript

Copy code

db.Book.find({ publisher\_name: "Pearson" });

17. Create a collection named Book. (book\_isbn,title,punlisher\_name,author(Name, Address, Phone No[landline, mobile]), publisher\_city, price,copies) a. Count the number of documents in the collection. b. Arrange the documents in descending order of book\_isbn. c. Select Book Names whose title is ”DBMS” . d. Update Book Copies as “10” whose Book Publisher is “Tata MacGraw Hill”. Display name of publishers as per no of books published by them in ascending order.

**ChatGPT said:**

ChatGPT

Here’s how you can create a MongoDB collection named Book and perform the specified operations:

**Step 1: Create the Collection and Insert Documents**

Assuming you already have the Book collection set up, here’s an example of how you can insert documents into it:

javascript

Copy code

use yourDatabaseName; // Replace with your actual database name

db.Book.insertMany([

{

book\_isbn: "978-0070634174",

title: "DBMS",

publisher\_name: "Tata MacGraw Hill",

author: {

name: "Rajesh",

address: "Mumbai, India",

phone: {

landline: "022-12345678",

mobile: "9876543210"

}

},

publisher\_city: "Mumbai",

price: 45.00,

copies: 5

},

{

book\_isbn: "978-0136006633",

title: "Introduction to Algorithms",

publisher\_name: "MIT Press",

author: {

name: "Thomas H. Cormen",

address: "Cambridge, MA",

phone: {

landline: "123-456-7890",

mobile: "987-654-3210"

}

},

publisher\_city: "Cambridge",

price: 59.99,

copies: 10

},

// Add more documents as needed

]);

**Step 2: Execute the Queries**

**a. Count the Number of Documents in the Collection**

javascript

Copy code

db.Book.countDocuments();

**b. Arrange the Documents in Descending Order of book\_isbn**

javascript

Copy code

db.Book.find().sort({ book\_isbn: -1 });

**c. Select Book Names Whose Title is "DBMS"**

javascript

Copy code

db.Book.find({ title: "DBMS" }, { title: 1, \_id: 0 });

**d. Update Book Copies as “10” Where Publisher is “Tata MacGraw Hill”**

javascript

Copy code

db.Book.updateMany(

{ publisher\_name: "Tata MacGraw Hill" },

{ $set: { copies: 10 } }

);

**e. Display Names of Publishers as Per Number of Books Published by Them in Ascending Order**

To get the count of books published by each publisher and sort them in ascending order, you can use the aggregation framework:

javascript

Copy code

db.Book.aggregate([

{ $group: { \_id: "$publisher\_name", count: { $sum: 1 } } },

{ $sort: { count: 1 } }

]);

18. . Create a collection named “ORDERS” that contain documents of the following prototype and solve the following queries: { cust\_id: &quot;abc123&quot;, ord\_date: new Date(&quot;Oct 04, 2012&quot;), status: &#39;A&#39;, price: 50, items: [ { sku: &quot;xxx&quot;, qty: 25, price: 1 }, { sku: &quot;yyy&quot;, qty: 25, price: 1 } ] } a. Count all records from orders b. Sum the price field from orders c. For each unique cust\_id, sum the price field. d. For each unique cust\_id, sum the price field, results sorted by sum. For each unique cust\_id, ord\_date grouping, sum the price field

**ChatGPT said:**

ChatGPT

Here's how you can create a MongoDB collection named ORDERS and perform the specified queries based on the provided document prototype.

**Step 1: Create the Collection and Insert Documents**

First, ensure you have a MongoDB database set up. You can create the ORDERS collection and insert sample documents as follows:

javascript

Copy code

use yourDatabaseName; // Replace with your actual database name

db.ORDERS.insertMany([

{

cust\_id: "abc123",

ord\_date: new Date("Oct 04, 2012"),

status: 'A',

price: 50,

items: [

{ sku: "xxx", qty: 25, price: 1 },

{ sku: "yyy", qty: 25, price: 1 }

]

},

{

cust\_id: "abc123",

ord\_date: new Date("Oct 05, 2012"),

status: 'A',

price: 75,

items: [

{ sku: "zzz", qty: 30, price: 2 }

]

},

{

cust\_id: "xyz789",

ord\_date: new Date("Oct 06, 2012"),

status: 'B',

price: 100,

items: [

{ sku: "aaa", qty: 50, price: 2 }

]

},

{

cust\_id: "xyz789",

ord\_date: new Date("Oct 07, 2012"),

status: 'A',

price: 150,

items: [

{ sku: "bbb", qty: 75, price: 2 }

]

},

{

cust\_id: "abc123",

ord\_date: new Date("Oct 08, 2012"),

status: 'C',

price: 120,

items: [

{ sku: "ccc", qty: 60, price: 2 }

]

}

]);

**Step 2: Execute the Queries**

**a. Count All Records from ORDERS**

javascript

Copy code

db.ORDERS.countDocuments();

**b. Sum the Price Field from ORDERS**

javascript

Copy code

db.ORDERS.aggregate([

{ $group: { \_id: null, totalPrice: { $sum: "$price" } } }

]);

**c. For Each Unique cust\_id, Sum the Price Field**

javascript

Copy code

db.ORDERS.aggregate([

{ $group: { \_id: "$cust\_id", totalPrice: { $sum: "$price" } } }

]);

**d. For Each Unique cust\_id, Sum the Price Field, Results Sorted by Sum**

javascript

Copy code

db.ORDERS.aggregate([

{ $group: { \_id: "$cust\_id", totalPrice: { $sum: "$price" } } },

{ $sort: { totalPrice: 1 } } // Change to -1 for descending order

]);

**e. For Each Unique cust\_id, ord\_date Grouping, Sum the Price Field**

javascript

Copy code

db.ORDERS.aggregate([

{ $group: { \_id: { cust\_id: "$cust\_id", ord\_date: "$ord\_date" }, totalPrice: { $sum: "$price" } } }

]);

19. Create a collection named rating that contain 5 documents of the following prototype and solve the following Queries. { movie\_id: 123, user\_id: 12, title: Toy Story(1995), status: &#39;A&#39; } a) Creating an index on movie\_id and sorts the keys in the index in ascending order. Verify the query plan b) Show various indexes created on movie collection. c) Sort movie\_id in descending order. d) Create a descending order index on movie\_id to get ratings related to “Toy Story (1995)” verify the query plan. e) Limit the number of items in the result of above query.

**ChatGPT said:**

ChatGPT

Here's how to create a MongoDB collection named rating, insert documents, and perform the specified queries.

**Step 1: Create the Collection and Insert Documents**

First, make sure you have a MongoDB database set up. You can create the rating collection and insert sample documents as follows:

javascript

Copy code

use yourDatabaseName; // Replace with your actual database name

db.rating.insertMany([

{ movie\_id: 123, user\_id: 1, title: "Toy Story (1995)", status: 'A' },

{ movie\_id: 124, user\_id: 2, title: "Finding Nemo (2003)", status: 'A' },

{ movie\_id: 123, user\_id: 3, title: "Toy Story (1995)", status: 'B' },

{ movie\_id: 125, user\_id: 4, title: "The Incredibles (2004)", status: 'A' },

{ movie\_id: 126, user\_id: 5, title: "WALL-E (2008)", status: 'A' }

]);

**Step 2: Execute the Queries**

**a) Creating an Index on movie\_id in Ascending Order**

javascript

Copy code

db.rating.createIndex({ movie\_id: 1 });

**Verify the Query Plan:**

You can check the query plan for a query using the explain() method:

javascript

Copy code

db.rating.find({ movie\_id: 123 }).explain("executionStats");

**b) Show Various Indexes Created on the rating Collection**

javascript

Copy code

db.rating.getIndexes();

**c) Sort movie\_id in Descending Order**

javascript

Copy code

db.rating.find().sort({ movie\_id: -1 });

**d) Create a Descending Order Index on movie\_id to Get Ratings Related to “Toy Story (1995)”**

javascript

Copy code

db.rating.createIndex({ movie\_id: -1 });

**Verify the Query Plan:**

javascript

Copy code

db.rating.find({ title: "Toy Story (1995)" }).explain("executionStats");

**e) Limit the Number of Items in the Result of Above Query**

You can limit the results of the query for "Toy Story (1995)" to, for example, 1 item:

javascript

Copy code

db.rating.find({ title: "Toy Story (1995)" }).limit(1);

23. Create a collection named Book. Add 5 documents in the collection with keys (book\_isbn,title,punlisher\_name,author(Name, Address, Phone No[landline, mobile]), publisher\_city, price,copies) a) Select Book Names whose title is ”DBMS” . b) Update Book Copies as “10” whose Book Publisher is “Tata MacGraw Hill”. c) Display name of publishers as per no of books published by them in ascending order. d) Get publisher names who published at least one book written by author name like ‘K%’. e) Delete the book from Book table written by Author ‘Korth’.

**ChatGPT said:**

ChatGPT

Here’s how to create a MongoDB collection named Book, add documents, and perform the specified queries.

**Step 1: Create the Collection and Insert Documents**

First, ensure you have a MongoDB database set up. You can create the Book collection and insert sample documents as follows:

javascript

Copy code

use yourDatabaseName; // Replace with your actual database name

db.Book.insertMany([

{

book\_isbn: "978-0070634174",

title: "DBMS",

publisher\_name: "Tata MacGraw Hill",

author: {

name: "Korth",

address: "Mumbai, India",

phone: {

landline: "022-12345678",

mobile: "9876543210"

}

},

publisher\_city: "Mumbai",

price: 45.00,

copies: 5

},

{

book\_isbn: "978-0136006633",

title: "Introduction to Algorithms",

publisher\_name: "MIT Press",

author: {

name: "Thomas H. Cormen",

address: "Cambridge, MA",

phone: {

landline: "123-456-7890",

mobile: "987-654-3210"

}

},

publisher\_city: "Cambridge",

price: 59.99,

copies: 10

},

{

book\_isbn: "978-0131101630",

title: "The C Programming Language",

publisher\_name: "Prentice Hall",

author: {

name: "Brian W. Kernighan",

address: "Pune, India",

phone: {

landline: "012-345-6789",

mobile: "987-654-3211"

}

},

publisher\_city: "Pune",

price: 45.00,

copies: 8

},

{

book\_isbn: "978-0201633610",

title: "Design Patterns",

publisher\_name: "Addison-Wesley",

author: {

name: "Erich Gamma",

address: "Boston, MA",

phone: {

landline: "234-567-8901",

mobile: "876-543-2109"

}

},

publisher\_city: "Boston",

price: 40.00,

copies: 8

},

{

book\_isbn: "978-0132350884",

title: "Clean Code",

publisher\_name: "Prentice Hall",

author: {

name: "Robert C. Martin",

address: "Chicago, IL",

phone: {

landline: "345-678-9012",

mobile: "765-432-1098"

}

},

publisher\_city: "Chicago",

price: 50.00,

copies: 7

}

]);

**Step 2: Execute the Queries**

**a) Select Book Names Whose Title is “DBMS”**

javascript

Copy code

db.Book.find({ title: "DBMS" }, { title: 1, \_id: 0 });

**b) Update Book Copies to “10” Where Publisher is “Tata MacGraw Hill”**

javascript

Copy code

db.Book.updateMany(

{ publisher\_name: "Tata MacGraw Hill" },

{ $set: { copies: 10 } }

);

**c) Display Name of Publishers as Per Number of Books Published by Them in Ascending Order**

You can use the aggregation framework to count books by publisher and sort the results:

javascript

Copy code

db.Book.aggregate([

{ $group: { \_id: "$publisher\_name", count: { $sum: 1 } } },

{ $sort: { count: 1 } }

]);

**d) Get Publisher Names Who Published at Least One Book Written by Author Name Like ‘K%’**

javascript

Copy code

db.Book.distinct("publisher\_name", { "author.name": { $regex: /^K/i } });

**e) Delete the Book from the Collection Written by Author ‘Korth’**

javascript

Copy code

db.Book.deleteMany({ "author.name": "Korth" });

24. onsider following structure for MongoDB collections and write a query for following requirements in MongoDB Teachers(Tname, dno, experience, salary, date\_of joining) Students(Sname, roll\_no, class) i) Write a MongoDB query to create above collections &amp; for insertion of some sample documents. ii) Find the information about all teachers of dno = 2 and having salary greater than or equal to 10,000/- iii) Find the student information having roll\_no = 2 or Sname = Anil iv) Display Total no of Students of TE Class V) update salary as 5% increment of teacher whose experience is &gt;10 years.

**ChatGPT said:**

ChatGPT

Here's how you can create the MongoDB collections for Teachers and Students, insert sample documents, and execute the specified queries.

**Step 1: Create Collections and Insert Sample Documents**

First, ensure you have a MongoDB database set up. You can create the Teachers and Students collections and insert sample documents as follows:

javascript

Copy code

use yourDatabaseName; // Replace with your actual database name

// Create Teachers collection and insert sample documents

db.Teachers.insertMany([

{ Tname: "John Doe", dno: 2, experience: 12, salary: 15000, date\_of\_joining: new Date("2010-05-01") },

{ Tname: "Jane Smith", dno: 1, experience: 8, salary: 9000, date\_of\_joining: new Date("2013-08-15") },

{ Tname: "Alice Johnson", dno: 2, experience: 15, salary: 18000, date\_of\_joining: new Date("2008-03-20") },

{ Tname: "Bob Brown", dno: 3, experience: 5, salary: 7500, date\_of\_joining: new Date("2016-06-30") }

]);

// Create Students collection and insert sample documents

db.Students.insertMany([

{ Sname: "Anil", roll\_no: 1, class: "TE" },

{ Sname: "Ravi", roll\_no: 2, class: "BE" },

{ Sname: "Sneha", roll\_no: 3, class: "TE" },

{ Sname: "Raj", roll\_no: 4, class: "SE" }

]);

**Step 2: Execute the Queries**

**i) Query for Teachers with dno = 2 and Salary ≥ 10,000**

javascript

Copy code

db.Teachers.find({ dno: 2, salary: { $gte: 10000 } });

**ii) Query for Student Information with roll\_no = 2 or Sname = "Anil"**

javascript

Copy code

db.Students.find({ $or: [{ roll\_no: 2 }, { Sname: "Anil" }] });

**iii) Display Total Number of Students in TE Class**

javascript

Copy code

db.Students.countDocuments({ class: "TE" });

**iv) Update Salary by 5% for Teachers with Experience > 10 Years**

javascript

Copy code

db.Teachers.updateMany(

{ experience: { $gt: 10 } },

{ $mul: { salary: 1.05 } }

);