Lab Question bank or Question pattern SQL statements, Hadoop and java map reduce operations and Mogo DB

SQL statements

1. Create the following tables under MCIS_your Regester number database

branch-name	account-number	balance
Downtown	A-101	500
Mianus	A-215	700
Perrryridge	A-102	400
Round Hill	A-305	350
Brighton	A-201	900
Redwood	A-222	700
Brighton	A-217	750
Account relation		

Customer-name	Customer-street	Customer-city
Jones	Main	Harrison
Smith	North	Rye
Hayes	Main	Harrison
Curry	North	Rye
Lindsay	Park	Pittsfield
Turner	Putnam	Stamford
Williams	Nassau	Princeton
Adams	Spring	Pittsfield
Johnson	Alma	Palo Alto
Glenn	Sand Hill	Woodside
Brooks	Senator	Brooklyn
Green	Walnut	Stamford
customer relation		

Customer-name	Loan-Number
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Currry	L-93
Smith	L-11
Williams	L-17
Adams	L-16
borower relation	

branch-name	branch-city	balance
Downtown	Brooklyn	9000000
Redwood	Palo Alto	2100000
Perrryridge	Horseneck	1700000
Mianus	Horseneck	400000
Round Hill	Horseneck	8000000
Pownal	Bennington	3000000
Northton	Rye	3700000
Brighton	Brooklyn	7100000
Branch relation		

Customer-name	Account-Number
Jones	A-101
Smith	A-215
Hayes	A-102
Turner	A-305
Johnson	A-201
Jones	A-217
Lindsay	A-222
depositor relation	

branch-name	Loan - number	amount
Downtown	L-17	1000
Redwood	L-23	2000
Perrryridge	L-15	1500
Downtown	L-14	1500
Mianus	L-93	500
Round Hill	L-11	900
Perrryridge	L-17	1300
Loan relation		

Set 1

1. Create branch table and Declare *branch_name* as the primary key for *branch, and branch_city should not take NULL values*
2. Add a new tuple to *account with values 'A-9732', 'Perryridge', 1200*
3. Use The **alter table** command to add new attribute PhoneNumber to an existing relation customer
4. Use The **drop table** command to remove the new attribute column PhoneNumber from relation customer

5. Find the names of all branches in the *loan* relation and remove duplicates.

Set 2

6. Find the names of all branches in the *loan* relation and do not remove duplicates.
7. Display all the contents of the table without mentioning names of the attributes
8. Multiply amount attribute with value 100 for all the loan numbers in the loan
9. Find all loans over $1200
10. Find the loan number for each loan of an amount > $1200

Set 3

11. Provide as a gift for all loan customers of the Perryridge branch, a $200 savings account. Let the loan number serve as the account number for the new savings account
12. Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%. Write two **update** statements:
13. Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.(Use case statement)
14. Find all customers who have at least two accounts at the Perryridge branch.
15. Write the SQL queries for the operations below using the relations loan and borrower given below?

- Relation *loan*

loan-number	branch-name	amount
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relation borrower

customer-name	loan-number
Jones	L-170
Smith	L-230
Hayes	L-155

i. Natural join
ii. Left Outer join
iii. Right outer join

Hadoop and java map reduce operations

MapReduce Questions

Create dataset with fields like 'Student Name', 'Institute', 'Program Name', and 'Gender'

and solve following questions.

1. Compute number of students from each Institute.
2. Number of students enrolled to any program.
3. Number of ‘boy’ and ‘girl’ students.
4. Number of ‘boy’ and ‘girl’ students from selected Institute.

Dataset: Temperature of Indian Cities. Fields of dataset are Date, Average Temperature, City, Country, Latitude and Longitude (Dataset is attached). Solve following questions

1. Find maximum and minimum temperature of all cities from the given dataset
2. Count number of data point for each city.
3. Find the maximum and minimum temperature for city **Bangalore** from the given dataset.
4. Find the maximum and minimum temperature for any given city from the given dataset. City name should be passed through command line argument.

Mogo DB

1. Insert a Single Document

Demonstrate insert of the single document into the `inventory` collection
{ item: **"canvas"**, qty: 100, tags: [**"cotton"**], size: { h: 28, w: 35.5, uom: **"cm"** } }

Ans:

```
db.inventory.insertOne(
  { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
)
```

Demonstrate insert of the single document into the `inventory` collection ?

{ item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }

{item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },

{ item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }

Demonstrate the retrieve the document that you just inserted

Ans

```
db.inventory.find( { item: "canvas" } )
```

Insert Multiple Documents

Demonstrate the insert of three new documents into the `inventory` collection

```
{ item: "journal", qty: 25, tags: ["blank", "red"], size: { h:
14, w: 21, uom: "cm" } },
   { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w:
35.5, uom: "cm" } },
   { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h:
19, w: 22.85, uom: "cm" } }
```

Ans

db.inventory.insertMany([

{ item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },

{ item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },

{ item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }

])

Demonstrate the retrieve of multiple documents that you just inserted

Ans

```
db.inventory.find( {} )
```

Query Documents

The examples on this page use the `inventory` collection. To populate the `inventory` collection, run the following:

1. populate the following `inventory` collection?

{ item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },

{ item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },

{ item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },

{ item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },

{ item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }

Ans

```
db.inventory.insertMany([
  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
  { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

To specify equality conditions, use <field>:<value> expressions in the query filter document:

2. Select from the inventory collection all documents where the status equals "D"?

Ans -

```
db.inventory.find( { status: "D" } )
```

3. Select from the inventory collection all documents where the status equals "A"?
4. select from the inventory collection all documents where the qty equals "50"?

Ans -

```
db.inventory.find( { qty: 50 } )
```

5. select from the inventory collection all documents where the qty equals "45"?
6. select from the inventory collection all documents where the qty equals "25"?
7. select from the inventory collection all documents where the qty equals "100"?
8. select from the inventory collection all documents where the qty equals "75"?
9. Retrieves all documents from the `inventory` collection where `status` equals either `"A"` or `"D"`

Ans -

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } )
```

10. Retrieves all documents from the `inventory` collection where `qty` equals either `50` or `25`
11. Retrieves all documents from the `inventory` collection where `qty` equals either `50` or `25`
12. Retrieves all documents from the `inventory` collection where `qty` equals either `100` or `75`
13. Retrieves all documents from the `inventory` collection where `qty` equals either 2`5` or `75`

Retrieves all documents in the `inventory` collection where the `status` equals `"A"` **and** `qty` is less than (`$lt`) `30`:

Ans –

```
db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )
```

14. Retrieves all documents in the inventory collection where the status equals "D" **and** qty is less than ($lt) 100:
15. Retrieves all documents in the inventory collection where the status equals "D" **and** qty is less than ($gt) 20:
16. Retrieves all documents in the inventory collection where the status equals "A" **and** qty is less than ($lt) 60:
17. Retrieves all documents in the inventory collection where the status equals "A" **and** qty is less than ($lt) 50:

Specify `AND` as well as `OR` Conditions

the compound query document selects all documents in the collection where the `status` equals `"A"` **and** *either* `qty` is less than (`$lt`) `30` *or* `item` starts with the character `p`:

Ans –

```
db.inventory.find( {
     status: "A",
     $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
} )
```

18. the compound query document selects all documents in the collection where the `status` equals `"D"` **and** *either* `qty` is less than (`$lt`) `101` *or* `item` starts with the character `p`:
19. write the compound query document selects all documents in the collection where the `status` equals `"D"` **and** *either* `qty` is less than (`$lt`) `100` *or* `item` starts with the character `p`:

Match an Embedded/Nested Document

To specify an equality condition on a field that is an embedded/nested document, use the query filter document { <field>: <value> } where <value> is the document to match.

For example, the following query selects all documents where the field size equals the document { h: 14, w: 21, uom: "cm" }:

db.inventory.find({ size: { h: 14, w: 21, uom: "cm" } })

Equality matches on the whole embedded document require an exact match of the specified <value> document, including the field order. For example, the following query does not match any documents in the inventory collection:

db.inventory.find({ size: { w: 21, h: 14, uom: "cm" } })

20. write query selects all documents where the field size equals the document { h: 8.5, w: 11, uom: "in"}
21. write query selects all documents where the field size equals the document { h: 22.85, w: 30, uom: "cm"}

Match an Array

22. populate the `inventory` collection
    { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [14, 21] },
    { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [14, 21] },
    { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [14, 21] },
    { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [22.85, 30] },
    { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [10, 15.25] }

    Ans -

db.inventory.insertMany([

{ item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [14, 21] },

{ item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [14, 21] },

{ item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [14, 21] },

{ item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [22.85, 30] },

{ item: "postcard", qty: 45, tags: ["blue"], dim_cm: [10, 15.25] }

]);

Write query for all documents where the field `tags` value is an array with exactly two elements, `"red"` and `"blank"`, in the specified order:

Ans –

db.inventory.find({ tags: ["red", "blank"] })

23. Write query for all documents where the field `tags` value is an array with exactly two elements `"red"`, `"blank"` , and “plain” in the specified order:
24. Write query for all documents where the field `tags` value is an array with exactly three elements, `"blank"` and `"red"`, in the specified order:
25. Write query for all documents where the field `tags` value is an array with exactly three elements, `"blue"` , in the specified order:

find an array that contains both the elements `"red"` and `"blank"`, without regard to order or other elements in the array, use the `$all` operator:

Ans -

db.inventory.find({ tags: { $all: ["red", "blank"] } })

26. find an array that contains both the elements `"blue"` , without regard to order or other elements in the array, use the `$all` operator

Query an Array for an Element

To query if the array field contains at least *one* element with the specified value, use the filter `{ <field>: <value> }` where `<value>` is the element value.

The following example queries for all documents where `tags` is an array that contains the string `"red"` as one of its elements:

Ans –

```
db.inventory.find( { tags: "red" } )
```

27. Write query for all documents where tags is an array that contains the string "blank" as one of its elements:

Write query for all documents where tags is an array that contains the string "blue" as one of its elements:

If, instead, you wish to find an array that contains both the elements `"red"` and `"blank"`, without regard to order or other elements in the array, use the `$all` operator:

Ans –

db.inventory.find({ tags: { $all: ["red", "blank"] } })

28. find an array that contains both the elements `"blank"` and `"red"` , without regard to order or other elements in the array, use the `$all` operator:

Query an Array for an Element

The following example queries for all documents where `tags` is an array that contains the string `"red"` as one of its elements:

Ans -

```
db.inventory.find( { tags: "red" } )
```

29. write query for all documents where tags is an array that contains the string "blank" as one of its elements:
30. write query for all documents where tags is an array that contains the string "plain" as one of its elements:
31. write query for all documents where tags is an array that contains the string "red" as one of its elements:
32. write query for all documents where tags is an array that contains the string "blue" as one of its elements:

To specify conditions on the elements in the array field, use query operators in the query filter document:

For example, the following operation queries for all documents where the array `dim_cm` contains at least one element whose value is greater than `25`.

Ans

db.inventory.find({ dim_cm: { $gt: 25 } })

33. write query for, all documents where the array `dim_cm` contains at least one element whose value is greater than `25`.
34. write query for, all documents where the array `dim_cm` contains at least one element whose value is greater than `30`.

Specify Multiple Conditions for Array Elements

When specifying compound conditions on array elements, you can specify the query such that either a single array element meets these condition or any combination of array elements meets the conditions.

The following example queries for documents where the `dim_cm` array contains elements that in some combination satisfy the query conditions; e.g., one element can satisfy the greater than `15` condition and another element can satisfy the less than `20` condition, or a single element can satisfy both:

Ans -

```
db.inventory.find( { dim_cm: { $gt: 15, $lt: 20 } } )
```

35. write query for documents where the `dim_cm` array contains elements that in some combination satisfy the query conditions; e.g., one element can satisfy the greater than `15` condition and another element can satisfy the less than `100` condition, or a single element can satisfy both:

36. write query for documents where the `dim_cm` array contains elements that in some combination satisfy the query conditions; e.g., one element can satisfy the greater than `20` condition and another element can satisfy the less than `90` condition, or a single element can satisfy both:
37. write query for documents where the `dim_cm` array contains elements that in some combination satisfy the query conditions; e.g., one element can satisfy the greater than `20` condition and another element can satisfy the less than `90` condition, or a single element can satisfy both:

Query for an Array Element that Meets Multiple Criteria

Use `$elemMatch` operator to specify multiple criteria on the elements of an array such that at least one array element satisfies all the specified criteria.

The following example queries for documents where the `dim_cm` array contains at least one element that is both greater than (`$gt`) `22` and less than (`$lt`) `30`:

```
db.inventory.find( { dim_cm: { $elemMatch: { $gt: 22, $lt: 30 } } } )
```

38. write query for documents where the dim_cm array contains at least one element that is both greater than ($gt) 22 and less than ($lt) 30:
39. write query for documents where the dim_cm array contains at least one element that is both greater than ($gt) 10 and less than ($lt) 30:
40. write query for documents where the dim_cm array contains at least one element that is both greater than ($gt) 14 and less than ($lt) 21:

Query for an Element by the Array Index Position

Using dot notation, you can specify query conditions for an element at a particular index or position of the array. The array uses zero-based indexing.

Note - When querying using dot notation, the field and nested field must be inside quotation marks.

The following example queries for all documents where the second element in the array `dim_cm` is greater than `25`:

Ans -

```
db.inventory.find( { "dim_cm.1": { $gt: 25 } } )
```

41. Write Query for all documents where the second element in the array `dim_cm` is greater than `25`:
42. Write Query for all documents where the second element in the array `dim_cm` is greater than `14`:
43. Write Query for all documents where the second element in the array `dim_cm` is greater than `10`:
44. Write Query for all documents where the first element in the array `dim_cm` is greater than `10`:
45. Write Query for all documents where the first element in the array `dim_cm` is greater than `14`:
46. Write Query for all documents where the first element in the array `dim_cm` is greater than `20`:

Query an Array by Array Length

Use the `$size` operator to query for arrays by number of elements. For example, the following selects documents where the array `tags` has 3 elements.

```
db.inventory.find( { "tags": { $size: 3 } } )
```

47. Write Query Use the $size operator to selects documents where the array tags has 3 elements.
48. Write Query Use the $size operator to selects documents where the array tags has 2 elements.
49. Write Query Use the $size operator to selects documents where the array tags has 1 element.

populate the `inventory` collection

{ item: "journal", instock: [{ warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 }] },

{ item: "notebook", instock: [{ warehouse: "C", qty: 5 }] },

{ item: "paper", instock: [{ warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 }] },

{ item: "planner", instock: [{ warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5 }] },

{ item: "postcard", instock: [{ warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 }]
}

Ans –

db.inventory.insertMany([

{ item: "journal", instock: [{ warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 }] },

{ item: "notebook", instock: [{ warehouse: "C", qty: 5 }] },

{ item: "paper", instock: [{ warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 }] },

{ item: "planner", instock: [{ warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5 }] },

{ item: "postcard", instock: [{ warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 }]
}

]);

Use the Array Index to Query for a Field in the Embedded Document

Using dot notation, you can specify query conditions for field in a document at a particular index or position of the array. The array uses zero-based indexing.

NOTE

When querying using dot notation, the field and index must be inside quotation marks.

The following example selects all documents where the `instock` array has as its first element a document that contains the field `qty` whose value is less than or equal to `20`:

```
db.inventory.find( { 'instock.0.qty': { $lte: 20 } } )
```

50. selects all documents where the `instock` array has as its first element a document that contains the field `qty` whose value is less than or equal to `20`:
51. selects all documents where the `instock` array has as its first element a document that contains the field `qty` whose value is less than or equal to `35`:

A Single Nested Document Meets Multiple Query Conditions on Nested Fields

52. Use `$elemMatch` operator to specify multiple criteria on an array of embedded documents such that at least one embedded document satisfies all the specified criteria.

queries for documents where the `instock` array has at least one embedded document that contains both the field `qty` equal to `5` and the field `warehouse` equal to `A`:

Ans-

db.inventory.find({ "instock": { $elemMatch: { qty: 5, warehouse: "A" } } })

Query for documents where the `instock` array has at least one embedded document that contains the field `qty` that is greater than `10` and less than or equal to `20`:

Ans-

db.inventory.find({ "instock": { $elemMatch: { qty: { $gt: 10, $lte: 20 } } } })