Name : Niriksha Shetty

USN : 201046046

ME(BDA) Assignment Questions for MDB-5252 lab June 2022

1. Create the following tables under MCIS_your Regester number database

   Create database MCIS_201046046;

Write SQL statements for the following

Set 1

1. Create branch table and Declare branch_name as the primary key for branch, and branch_city should not take NULL values

   Create table branch(

   Branch_name NOT NULL PRIMARY KEY,

   Branch_city VARCHAR(255) NOT NULL,

   Balance VARCHAR(255) NOT NULL)

2. Add a new tuple to account with values 'A-9732', 'Perryridge', 1200

   Insert into account values(‘A-9372’ , ‘Perryridge’, 1200)

3. Use The alter table command to add new attribute PhoneNumber to an existing relation customer

   ALTER TABLE Customer ADD Phone Number VARCHAR(20);

4. Use The drop table command to remove the new attribute column PhoneNumber from relation customer

   ALTER TABLE Customer DROP Column Phone Number;

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

   Select distinct branch_name

```
From loan
```

Set 2

6. Find the names of all branches in the loan relation and do not remove duplicates.

```
Select all branch_name
 From loan
```

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

```
Select loan-number, branch-name, amount * 100 from loan
```

9. Find all loans over $1200

```
Select amount
from loan
Where amount > 1200
```

10. Find the loan number for each loan of an amount > $1200

```
select loan_number
from loan
where 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

```
insert into account
```

select loan-number, branch-name, 200

from loan where branch-name = ‘Perryridge’

insert into depositor select customer-name, loan-number from loan, borrower where branch-name = ‘Perryridge’ and loan.account-number=borrower.account-number

12. Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.

Write two update statements:

update account set balance = balance _ 1.06

where balance > 10000

update account set balance = balance _ 1.05

where balance _ 10000

13. Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.

update account set balance = balance _ 1.06

where balance > 10000

update account set balance = balance _ 1.05

where balance _ 10000

14. Find all customers who have at least two accounts at the Perryridge branch.

select distinct T.customer-name

from depositor T

where not unique (select R.customer-name

from account, depositor as R where T.customer-name = R.customer-name

and R.account-number = account.account-number

and account.branch-name = ‘Perryridge’)

15. Write the SQL queries for the operations below using the relations loan and borrower given below?

Natural Join: Natural join does not utilize any of the comparison operators. In this type of join, the attributes should have the same name and domain. In Natural Join, there should be at least one common attribute between two relations. It performs selection forming equality on those attributes which appear in both relations and eliminates the duplicate attributes.

Select * from loan

Natural join borrower

Right join: The right join keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

Select branch-name, amount

From loan l1

Right join borrower b1

On l1.loan-number = b1.loan-number

Left join : The left join keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

Select branch-name, amount

From loan l1

left join borrower b1

On l1.loan-number = b1.loan-number

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.

```
public class studentData {

    public static class testMapper extends
Mapper<LongWritable,Text,Text,IntWritable>
    {
        public void map(LongWritable key, Text values, Context context) throws
IOException, InterruptedException
        {
            private Text Institute = new Text();
            private Text Program = new Text();
            private Text Gender = new Text();
            private Text Gen_Ins = new Text();
            String line = new String(values.toString());
            String student[] = line.split(" ");
            if(student[1].equals("Nitte") || student[1].equals("MSIS"))
            {
                Institute.set(student[1]);
                context.write(Institute,new IntWritable(1));
            }
            if(student[2].equals("BDA") || student[2].equals("AIML"))
            {
                Program.set(student[1]);
                context.write(Program,new IntWritable(1));
            }
            if(student[3].equals("Boy") || student[3].equals("Girl"))
            {
                Gender.set(student[3]);
                context.write(Gender,new IntWritable(1));
            }
            if(student[1].equals("Nitte") && student[3].equals("Boy") )||
(student[1].equals("Nitte") && student[3].equals("Girl"))
            {
                Gen_Ins.set(student[4]);
         context.write(new Text("Count of students in NITTE - "+Gen_Ins), new
IntWritable(1));
            }
        }
    }
```

2. Number of students enrolled to any program.

```
public class studentData {

   public static class testMapper extends
Mapper<LongWritable,Text,Text,IntWritable>
   {
       public void map(LongWritable key, Text values, Context context) throws
IOException, InterruptedException
       {
           private Text Institute = new Text();
           private Text Program = new Text();
           private Text Gender = new Text();
           private Text Gen_Ins = new Text();
           String line = new String(values.toString());
           String student[] = line.split(" ");
           if(student[1].equals("Nitte") || student[1].equals("MSIS"))
           {
              Institute.set(student[1]);
              context.write(Institute,new IntWritable(1));
           }
           if(student[2].equals("BDA") || student[2].equals("AIML"))
           {
              Program.set(student[1]);
              context.write(Program,new IntWritable(1));
           }
           if(student[3].equals("Boy") || student[3].equals("Girl"))
           {
              Gender.set(student[3]);
              context.write(Gender,new IntWritable(1));
           }
           if(student[1].equals("Nitte") && student[3].equals("Boy") )||
(student[1].equals("Nitte") && student[3].equals("Girl"))
           {
              Gen_Ins.set(student[4]);
             context.write(new Text("Count of students enrolled to any program -
"+Gen_Ins), new IntWritable(1));
          }
       }
}
```

3. Number of ‘boy’ and ‘girl’ students.

```
public class studentData {

   public static class testMapper extends
Mapper<LongWritable,Text,Text,IntWritable>
   {
        public void map(LongWritable key, Text values, Context context) throws
IOException, InterruptedException
        {
            private Text Institute = new Text();
            private Text Program = new Text();
            private Text Gender = new Text();
            private Text Gen_Ins = new Text();
            String line = new String(values.toString());
            String student[] = line.split(" ");
            if(student[1].equals("Nitte") || student[1].equals("MSIS"))
            {
                Institute.set(student[1]);
                context.write(Institute,new IntWritable(1));
            }
            if(student[2].equals("BDA") || student[2].equals("AIML"))
            {
                Program.set(student[1]);
                context.write(Program,new IntWritable(1));
            }
            if(student[3].equals("Boy") || student[3].equals("Girl"))
            {
                Gender.set(student[3]);
                context.write(Gender,new IntWritable(1));
}
            if(student[1].equals("Nitte") && student[3].equals("Boy") )||
(student[1].equals("Nitte") && student[3].equals("Girl"))
            {
                Gen_Ins.set(student[4]);
               context.write(new Text("Number of boys and girls student-"+Gen_Ins), new
IntWritable(1));
           }
        }
  }
```

4. Number of ‘boy’ and ‘girl’ students from selected Institute.

```
      public class studentData {

    public static class testMapper extends
Mapper<LongWritable,Text,Text,IntWritable>
    {
        public void map(LongWritable key, Text values, Context context) throws
IOException, InterruptedException
        {
            private Text Institute = new Text();
            private Text Program = new Text();
            private Text Gender = new Text();
            private Text Gen_Ins = new Text();
            String line = new String(values.toString());
            String student[] = line.split(" ");
            if(student[1].equals("Nitte") || student[1].equals("MSIS"))
            {
               Institute.set(student[1]);
               context.write(Institute,new IntWritable(1));
            }
            if(student[2].equals("BDA") || student[2].equals("AIML"))
            {
               Program.set(student[1]);
               context.write(Program,new IntWritable(1));
            }
            if(student[3].equals("Boy") || student[3].equals("Girl"))
            {
               Gender.set(student[3]);
               context.write(Gender,new IntWritable(1));
         }
            if(student[1].equals("Nitte") && student[3].equals("Boy") )||
         (student[1].equals("Nitte") && student[3].equals("Girl"))
            {
               Gen_Ins.set(student[4]);
              context.write(new Text("Number of boys and girls student in MSIS-
         "+Gen_Ins), new IntWritable(1));
           }
        }
     }
```

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

Mongo DB

1. Write Queries 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: "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" } }

ANS:

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

2.Write Query to retrieve the document that you just inserted?

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

Insert Multiple Documents

1.Write Query to 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" } }])

2.Write Query to Demonstrate the retrieve of multiple documents that you just inserted?

ANS:

`db.inventory.find( {} )`

Query Documents

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:

1. Write Query to Select from the inventory collection all documents where the status equals "D"?

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

2. Write Query to Select from the inventory collection all documents where the status equals "A"?

   Ans: **db.inventory.find({ status: "A" })**

3. Write Query to select from the inventory collection all documents where the qty equals "50"?

   Ans**: db.inventory.find({ qty: 50}**

4. select from the inventory collection all documents where the qty equals "45"?

   Ans**: db.inventory.find({ qty: 45}**

5. select from the inventory collection all documents where the qty equals "25"?

   Ans**: db.inventory.find({ qty: 25}**

6. select from the inventory collection all documents where the qty equals "100"?

   Ans**: db.inventory.find({ qty: 100}**

7. select from the inventory collectionall documents where the qty equals "75"?

Ans: **db.inventory.find({ qty: 75}**

8. Retrieves all documents from the`inventory`collection where`status`equals either"`A`"or"`D`"

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

9. Write Query to Retrieve all documents from the `inventory` collection where `qty` equals either `50` or `25`

   Ans : db.inventory.find({ status: { $in: ["50", "25"] } })

10. Write Query to Retrieve all documents from the `inventory` collection where `qty` equals either `50` or `25`

    Ans : db.inventory.find({ status: { $in: ["50", "25"] } })

11.Write Query to Retrieve all documents from the `inventory` collection where `qty` equals either `100` or `75`

Ans : db.inventory.find({ status: { $in: ["100", "75"] } })

11. Write Query to Retrieve all documents from the `inventory` collection where `qty` equals either 2`5` or `75`

    Ans : db.inventory.find({ status: { $in: ["25", "75"] } })

12. Write Query to Retrieve all documents in the inventory collection where the status equals "D" **and** qty is less than ($lt) 100:

    Ans : db.inventory.find({ $or: [{ status: "D"}, { qty: { $lt: 100} }] })

13. Write Query to Retrieve all documents in the inventory collection where the status equals "D" **and** qty is less than ($gt) 20:

Ans : `db.inventory.find( { $or: [ { status: "D"}, { qty: { $lt: 100} } ] } )`

SpecifyANDas well asOR

Conditionsthe compound query document selects all documents in the collection where thestatusequals"A"andeitherqtyis less than ($lt)30oritemstarts with the characterp:

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

Match an Array

14.

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] }]);

15.

Ans :

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

16.

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 for an Array Element that Meets Multiple

CriteriaUse`$elemMatch`operator to specify multiple criteria on the elements of an array such that at least one array element satisfies all the specified criteria.The following example queries for documents where the`dim_cm`array contains at least one element that is both greater than (`$gt`)`22`and less than (`$lt`)`30`:`db`.inventory.find`( { dim_cm: { $elemMatch: { $gt: 22, $lt: 30} } } )`

`ans :` Ans -`db`.inventory.find`( { "dim_cm.1": { $gt: 21} } )`