

MEASI INSTITUTE OF INFORMATION TECHNOLOGY

(Approved by AICTE & Affiliated to University of Madras)

CHENNAI – 600 014



MEASI
Institute of
Information
Technology

MASTER OF COMPUTER APPLICATIONS

ACADEMIC YEAR 2025-2026

SEMESTER – I

Practical Record

Data Engineering and Management Lab
(435E1D)

REG. NO : _____

NAME : _____

BATCH : _____

MEASI INSTITUTE OF INFORMATION TECHNOLOGY
(Approved by AICTE & Affiliated to University of Madras)
CHENNAI- 600 014

MCA PRACTICAL

Data Engineering and Management Lab (435E1D)

Academic Year 2025-2026

Semester - I

NAME	:	CLASS	:
REG-NO	:	BATCH	:

This is to certify that this is the bonafide record of work done in the Computer Science Laboratory of **MEASI Institute of Information Technology**, submitted for the **University of Madras** Practical Examination held on at **MEASI Institute of Information Technology, Chennai-600 014**.

STAFF IN-CHARGE

HEAD OF THE DEPARTMENT

INTERNAL EXAMINER

EXTERNAL EXAMINER

DATE: _____

INDEX

S.No.	Date	Program	Page No.	Signature
1		To customize your application using Zoho CRM		
2		Write a script to create a MongoDB database and perform insert operation		
3		Write a MongoDB script to perform query operations		
4		Write a MongoDB Script to perform update operations		
5		Write a MongoDB Script to update documents with aggregation pipeline		
6		Write a MongoDB script to delete single and multiple documents		
7		Write a MongoDB script to perform string aggregation operations		
8		Design a Data Model for MongoDB using DbVisualizer		
9		Perform CRUD operations using DbVisualizer		
10		Create a Zoho CRM account and organize your Tasks, Meetings and Deals		
11		Create and maintain a project using Zoho CRM features		

Ex.No : 1 To customize your application using Zoho CRM

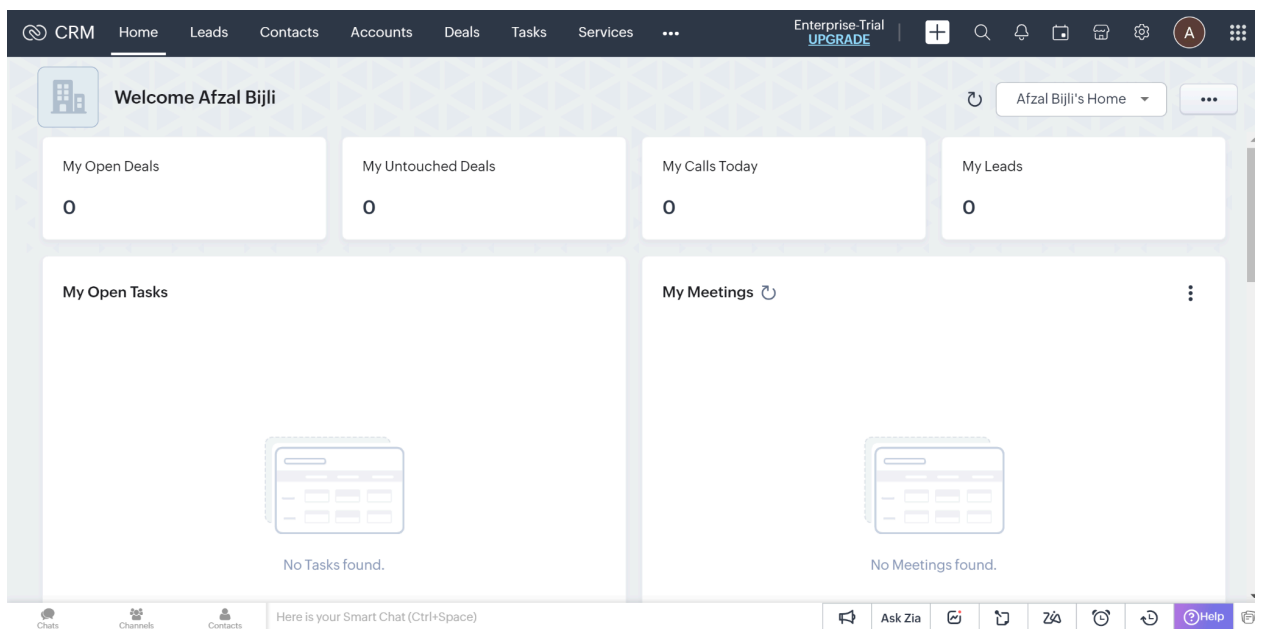
Date :

Aim

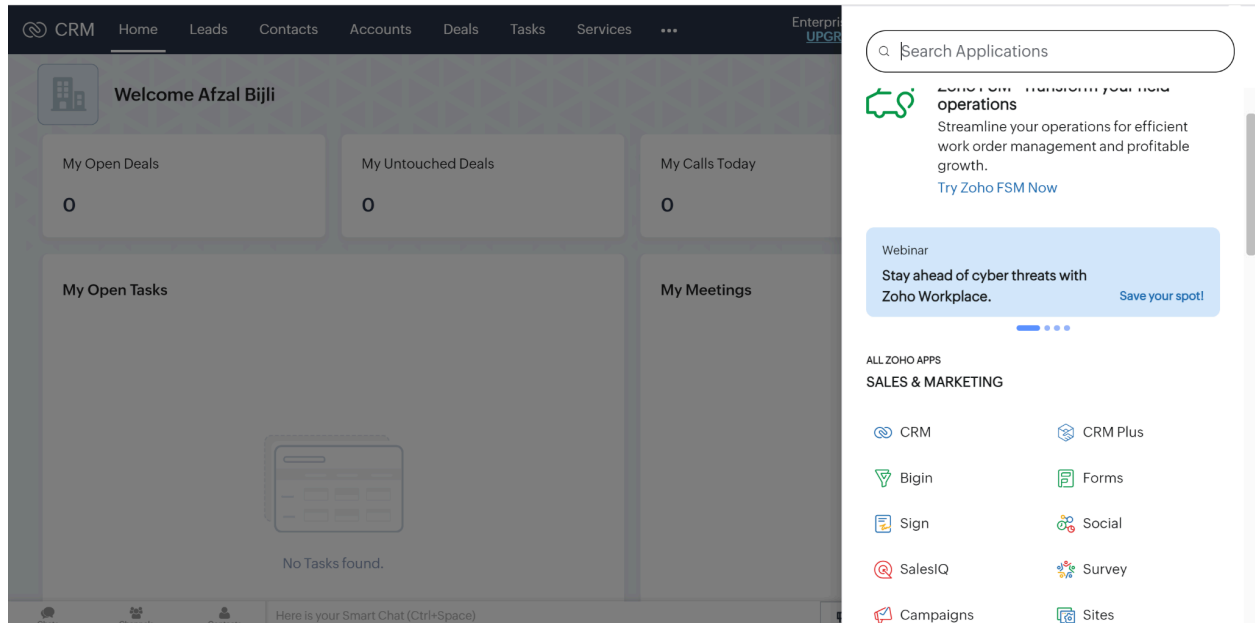
To customize your application using Zoho CRM

Algorithm

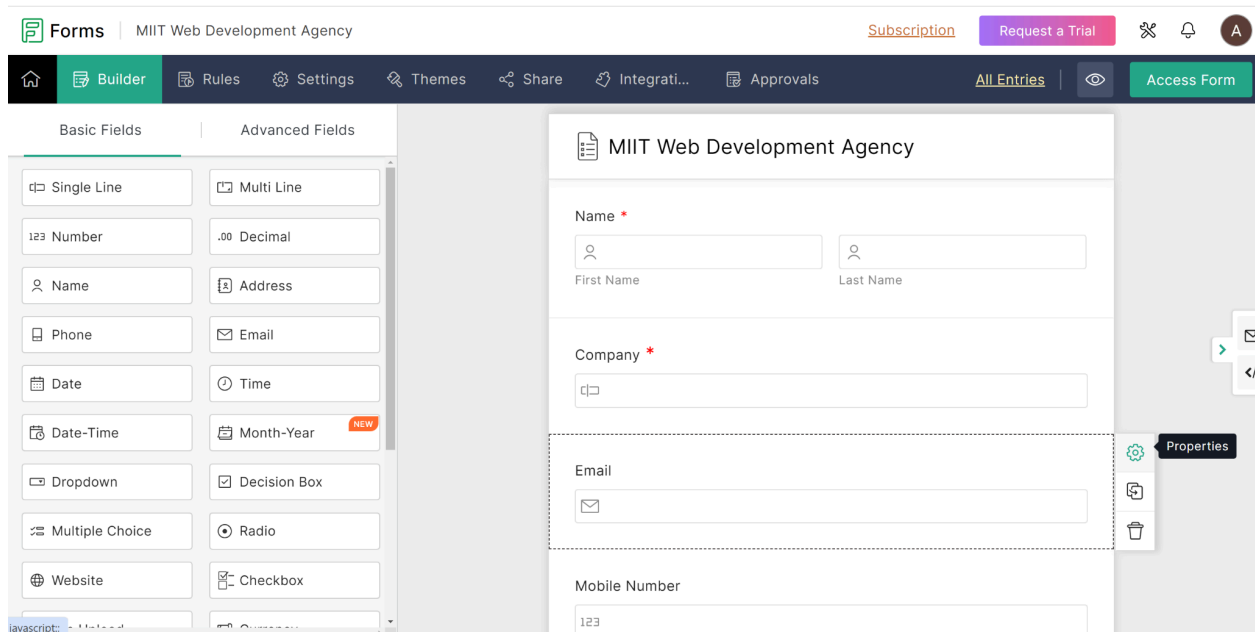
Step 1 : Goto <https://www.zoho.com/en-in/crm/> and signup for a new account



Step 2 : Select the Apps menu -> Sales & Marketing -> Forms -> New Form



Step 3 : Create a form using drag and drop elements with various fields such as Name, Company, Email id, Mobile Number & Services offered using dropdown menu



Step 4 : Click on preview form & select the theme

The image shows a preview of a form titled "MIIT Web Development Agency" in a stylized orange font. The form has a dark blue background with abstract circular patterns. It contains three input fields: a "Name" field with a red asterisk, which is split into "First Name" and "Last Name" sub-fields; and a "Company" field with a red asterisk. Each field has a small icon (person for name, building for company) and a placeholder text.

Step 5 : Click on integrate and do field mapping to successfully integrate the form into the lead module of Zoho CRM

The image is a screenshot of the Zoho Forms integration interface. The top navigation bar includes "Forms", "MIIT", "Subscription", "Request a Trial", and user icons. The main navigation bar has "Builder", "Rules", "Settings", "Themes", "Share", "Integrati...", and "Approvals". The "Integrations" sidebar on the left lists "Zoho CRM" as the selected integration, along with other options like "Begin by Zoho CRM", "Zoho SalesIQ", "Zoho Desk", "Zoho Campaigns", "Zoho Sheet", "Zoho Sign", and "Zoho Projects". The main content area is titled "Integrate with Zoho CRM" and shows "New Record" and "Related List" tabs. It displays the "Module" as "Leads" and "Layout" as "Standard". Below this, a "Field Mapping" section shows the mapping between Zoho Forms fields and Zoho CRM fields: "Company" maps to "Company", "Last Name" maps to "Name - Last Name", "First Name" maps to "Name - First Name", and "Email" maps to "Email". Each mapping is shown with a visual connector and a dropdown menu to select the target field.

Step 6 : Click on the Share form & copy link

The screenshot shows the Zoho Forms 'Share' interface. The top navigation bar includes 'Forms | MIT Web Development Agency', a 'Subscription' link, a 'Request a Trial' button, and user profile icons. The main menu has options like 'Builder', 'Rules', 'Settings', 'Themes', 'Share' (highlighted), 'Integrati...', and 'Approvals'. On the right, there are links for 'All Entries' and 'Access Form'.

The 'Share' panel is divided into two main sections. The left section, titled 'SHARE WITH', includes a 'Public' tab (selected), 'Specific Users', 'Groups', and 'All Users'. The right section, titled 'Share Publicly', has an 'Enabled' toggle. It contains a 'Download' button, a 'Track referrals' section with a URL template and a 'Know more' link, a 'Prefill form fields' section with another URL template and a 'Know more' link, and a 'Share on social media' section with icons for Facebook, Twitter, and LinkedIn.


Step 7 : Open the link in another browser or in incognito mode and fill the form

The screenshot shows a web browser window with the URL 'forms.zohopublic.in/afzalbijlimea1/form/WebDevelopment/formperma/G0Lp3fYLJHt5tvlm0RRqtbCYu5eT_RfzaKP0MPIDRtM'. The browser is in 'Incognito' mode. The form is titled 'MIT Web Development Agency' and has a dark blue background with circular patterns. It contains the following fields:

- Name ***: Two input fields for 'First Name' and 'Last Name'.
- Company ***: A single input field.
- Email**: A single input field.

At the bottom right of the form, there is a 'Desktop 2' button.






Step 8 : Once form is filled by the client, you get the details into form entry


 All Entries

New Report

All Entries ▾

List View ▾ | 🕒 | 📄 | 🔍 | ☰

	<input type="checkbox"/>	 Name	 Company	 Email	 Mobile Number	
	<input type="checkbox"/>	Rafath, Zahra	MEASI	rafath.zahra@measit.edu.in	1234567890	

 All times are in Asia/Kolkata

10 Records per page ▾

< 1 to 1 of 1 >

Result:

The customization of the application using Zoho CRM is successfully implemented.

Ex.No : 2 Write a script to create a MongoDB database and perform insert operation

Date :

Aim

To write a script to create a MongoDB database and perform insert operation

Algorithm

Step 1 : Open cmd prompt & type “mongosh” to connect with mongo db server

Step 2: Create a database & Collections

Step 3: Perform the insert operation

Step 4 : Display the output

Scripts Execution (Program)

1.To view databases

test> show dbs

```
admin> show dbs
admin      132.00 KiB
books      56.00 KiB
coll       296.00 KiB
```

2.Create New Database

test> use school

```
admin> use school
switched to db school
school> |
```

3.Create Collection

```
school> db.createCollection("students")
{ ok: 1 }
school> |
```

4.Show Collection Names

```
school> db.getCollectionNames()
```

```
school> db.getCollectionNames()
[ 'students' ]
school> |
```

5.Get the details of content in collection

```
school> db.getCollectionInfos()
```

```
school> db.getCollectionInfos()
[
  {
    name: 'students',
    type: 'collection',
    options: {},
    info: {
      readOnly: false,
      uuid: UUID('9b0c2043-5fff-4fd4-92a9-c14eaa9de9ca')
    },
    idIndex: { v: 2, key: { _id: 1 }, name: '_id_' }
  }
]
school> |
```

6.Insert the one document into the collection

```
school> db.students.insertOne({name:"afzal",age:20,gpa:8.9})
```

```
school> db.students.insertOne({name:"afzal",age:20,gpa:8.9})
{
  acknowledged: true,
  insertedId: ObjectId('67166cb7a48b3743a22710bf')
}
school> |
```

7.Fetch the details from the collection

school> db.students.find()

```
school> db.students.find()
[
  {
    _id: ObjectId('67166cb7a48b3743a22710bf'),
    name: 'afzal',
    age: 20,
    gpa: 8.9
  }
]
school> |
```

8.Insert more than one document into the collection

school>db.students.insertMany([{name:"karthik",age:20,gpa:3.4},{name:"rafaz",age:23,gpa:3.4,marks:77},{name:"kareem",age:24,gpa:3.4}])

```
school> db.students.insertMany([{name:"karthik",age:20,gpa:3.4},{name:"rafaz",age:23,gpa:3.4,marks:77},{name:"kareem",age:24,gpa:3.4}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67166d7fa48b3743a22710c0'),
    '1': ObjectId('67166d7fa48b3743a22710c1'),
    '2': ObjectId('67166d7fa48b3743a22710c2')
  }
}
school> |
```

9.Fetch the details from the collection

school> db.students.find()

```
school> db.students.find()
[
  {
    _id: ObjectId('67166cb7a48b3743a22710bf'),
    name: 'afzal',
    age: 20,
    gpa: 8.9
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c0'),
    name: 'karthik',
    age: 20,
    gpa: 3.4
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c1'),
    name: 'rafaz',
    age: 23,
    gpa: 3.4,
    marks: 77
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c2'),
    name: 'kareem',
    age: 24,
    gpa: 3.4
  }
]
```

Result

The Script to create a MongoDB database and perform insert operation is successfully implemented.

Ex.No : 3 Write a MongoDB script to perform query operations

Date :

Aim

To write a MongoDB script to perform query operations

Algorithm

Step 1 : Open cmd prompt & type “mongosh” to connect with mongo db server

Step 2: Create a database & Collections

Step 3: Perform the query operations

Step 4 : Display the output

Commands to remember

- **show dbs** - To view the available database
- **use databasename** - To go into a database
- **use test** - to come to the main
- **use newdatabasename** - To go into a database which was not created before
- **show collections** - To show the collections inside a Database
- **db.collectionname.find()** - to view documents
- **db.collectionname.insertOne({})** - To insert a Document
- **db.collectionname.insertMany([{}, {}, {}])** - To insert many documents in the collection

Scripts Execution (Program)

Note:

Here from the previous program let's assume we have already created a database named “school”, a collection named “students” & inserted four documents into it.

1.Filtering/Querying

- **db.collectionname.find({key:value, key:value })**

```

school> db.students.find()
[
  {
    _id: ObjectId('67166cb7a48b3743a22710bf'),
    name: 'afzal',
    age: 20,
    gpa: 8.9
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c0'),
    name: 'karthik',
    age: 20,
    gpa: 3.4
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c1'),
    name: 'rafaz',
    age: 23,
    gpa: 3.4,
    marks: 77
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c2'),
    name: 'kareem',
    age: 24,
    gpa: 3.4
  }
]

```

- **db.collectionname.find({key:value, key:value},{key:1})**

(Gives you only the details what you ask for in the second argument and not all the details after filtering based on the first argument)

```

school> db.students.find({age:20},{name:1})
[
  { _id: ObjectId('67166cb7a48b3743a22710bf'), name: 'afzal' },
  { _id: ObjectId('67166d7fa48b3743a22710c0'), name: 'karthik' }
]

```

- **db.collectionname.find({}, {key:1})**

(Gives you the only the details what you ask for and not all the details without any filter)

```

school> db.students.find({}, {name:1})
[
  { _id: ObjectId('67166cb7a48b3743a22710bf'), name: 'afzal' },
  { _id: ObjectId('67166d7fa48b3743a22710c0'), name: 'karthik' },
  { _id: ObjectId('67166d7fa48b3743a22710c1'), name: 'rafaz' },
  { _id: ObjectId('67166d7fa48b3743a22710c2'), name: 'kareem' }
]

```

- `db.collectionname.findOne({key:value})`

```
school> db.students.findOne({name:"rafaz"})
{
  _id: ObjectId('67166d7fa48b3743a22710c1'),
  name: 'rafaz',
  age: 23,
  gpa: 3.4,
  marks: 77
}
```

2.Method Chaining

(Following one method after another)

- `db.collectionname.find().count()`

```
school> db.students.find().count()
4
```

3.Sorting (Ascending & Descending)

- `db.collectionname.find().sort({key:1})` - Ascending

```
school> db.students.find().sort({age:1})
[
  {
    _id: ObjectId('67166cb7a48b3743a22710bf'),
    name: 'afzal',
    age: 20,
    gpa: 8.9
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c0'),
    name: 'karthik',
    age: 20,
    gpa: 3.4
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c1'),
    name: 'rafaz',
    age: 23,
    gpa: 3.4,
    marks: 77
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c2'),
    name: 'kareem',
    age: 24,
    gpa: 3.4
  }
]
```

- **db.collectionname.find().sort({key:-1})** - Descending

```
school> db.students.find().sort({age:-1})
[
  {
    _id: ObjectId('67166d7fa48b3743a22710c2'),
    name: 'kareem',
    age: 24,
    gpa: 3.4
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c1'),
    name: 'rafaz',
    age: 23,
    gpa: 3.4,
    marks: 77
  },
  {
    _id: ObjectId('67166cb7a48b3743a22710bf'),
    name: 'afzal',
    age: 20,
    gpa: 8.9
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c0'),
    name: 'karthik',
    age: 20,
    gpa: 3.4
  }
]
```

4.Operators in MongoDB

4.1 Comparison Operators

Name	Description
<code>\$eq</code>	Matches values that are equal to a specified value.
<code>\$gt</code>	Matches values that are greater than a specified value.
<code>\$gte</code>	Matches values that are greater than or equal to a specified value.
<code>\$in</code>	Matches any of the values specified in an array.
<code>\$lt</code>	Matches values that are less than a specified value.
<code>\$lte</code>	Matches values that are less than or equal to a specified value.
<code>\$ne</code>	Matches all values that are not equal to a specified value.
<code>\$nin</code>	Matches none of the values specified in an array.

- Example - Greater than operator (\$gt):

```
school> db.students.find({age:{$gt:20}})
[
  {
    _id: ObjectId('67166d7fa48b3743a22710c1'),
    name: 'rafaz',
    age: 23,
    gpa: 3.4,
    marks: 77
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c2'),
    name: 'kareem',
    age: 24,
    gpa: 3.4
  }
]
```

4.2 Logical Operators

Logical

Name	Description
<code>\$and</code>	Joins query clauses with a logical <code>AND</code> returns all documents that match the conditions of both clauses.
<code>\$not</code>	Inverts the effect of a query expression and returns documents that do <i>not</i> match the query expression.
<code>\$nor</code>	Joins query clauses with a logical <code>NOR</code> returns all documents that fail to match both clauses.
<code>\$or</code>	Joins query clauses with a logical <code>OR</code> returns all documents that match the conditions of either clause.

- Example - OR operator (\$or):

```
school> db.students.find({'$or': [{age:20},{age:23}]})
[
  {
    _id: ObjectId('67166cb7a48b3743a22710bf'),
    name: 'afzal',
    age: 20,
    gpa: 8.9
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c0'),
    name: 'karthik',
    age: 20,
    gpa: 3.4
  },
  {
    _id: ObjectId('67166d7fa48b3743a22710c1'),
    name: 'rafaz',
    age: 23,
    gpa: 3.4,
    marks: 77
  }
]
```

Result:

The MongoDB script to perform query operations is successfully executed

Ex.No : 4 Write a MongoDB Script to perform update operations

Date :

Aim

To Write a MongoDB Script to perform update operations

Algorithm

Step 1 : Open MongoDB Compass

Step 2: Create a database & Collections

Step 3: Insert the documents into it using “Add Data” option

Step 4: Open cmd prompt & type “mongosh” to connect with mongo db server

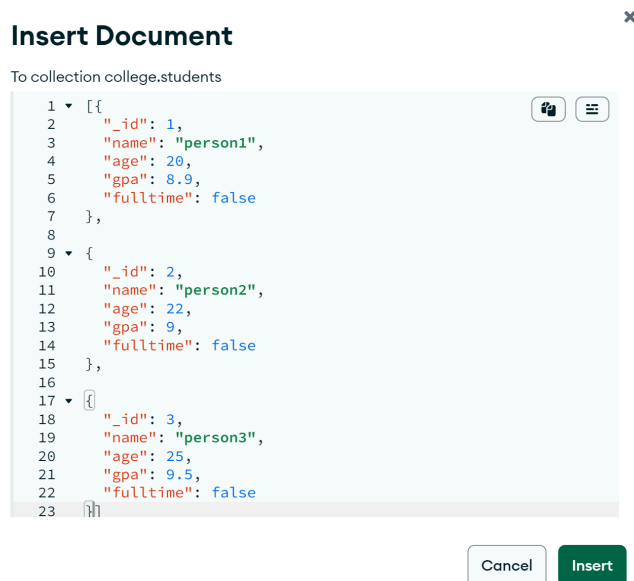
Step 5: Select the database which you created in compass

Step 6: Perform the update operations

Step 7 : Display the output

Scripts Execution (Program)

MongoDB Compass - Data Insertion



MongoDB Shell Commands

1. Select Database

admin > use college

```
admin> use college
switched to db college
```

college> show collections

```
college> show collections
students
college> db.students.find()
[
  { _id: 1, name: 'person1', age: 20, gpa: 8.9, fulltime: false },
  { _id: 2, name: 'person2', age: 22, gpa: 9, fulltime: false },
  { _id: 3, name: 'person3', age: 25, gpa: 9.5, fulltime: false }
]
```

college> db.students.find()

```
college> db.students.find()
[
  { _id: 1, name: 'person1', age: 20, gpa: 8.9, fulltime: false },
  { _id: 2, name: 'person2', age: 22, gpa: 9, fulltime: false },
  { _id: 3, name: 'person3', age: 25, gpa: 9.5, fulltime: false }
]
```

2. Update one value using \$set operator, here we are setting fulltime to true

college> db.students.updateOne({name:"person1"},{\$set:{fulltime:true}})

```
college> db.students.updateOne({name:"person1"},{$set:{fulltime:true}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

college> db.students.find({name:"person1"})

```
college> db.students.find({name:"person1"})
[ { _id: 1, name: 'person1', age: 20, gpa: 8.9, fulltime: true } ]
```

```
college> db.students.find()
[
  { _id: 1, name: 'person1', age: 20, gpa: 8.9, fulltime: true },
  { _id: 2, name: 'person2', age: 22, gpa: 9, fulltime: false },
  { _id: 3, name: 'person3', age: 25, gpa: 9.5, fulltime: false }
]
```

3.Update one value using \$set operator, here we are setting fulltime to false again

db.students.updateOne({_id:1},{ \$set:{fulltime:false}})

```
college> db.students.updateOne({_id:1},{ $set:{fulltime:false}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
college> db.students.find()
[
  { _id: 1, name: 'person1', age: 20, gpa: 8.9, fulltime: false },
  { _id: 2, name: 'person2', age: 22, gpa: 9, fulltime: false },
  { _id: 3, name: 'person3', age: 25, gpa: 9.5, fulltime: false }
]
```

3.Update one value using \$unset operator, here we are unsetting fulltime field from id:1

college> db.students.updateOne({_id:1},{ \$unset:{fulltime:""}})

```
college> db.students.updateOne({_id:1},{ $unset:{fulltime:""}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
college> db.students.find()
[
  { _id: 1, name: 'person1', age: 20, gpa: 8.9 },
  { _id: 2, name: 'person2', age: 22, gpa: 9, fulltime: false },
  { _id: 3, name: 'person3', age: 25, gpa: 9.5, fulltime: false }
]
```

4. Update Many values using \$set operator, here we are setting fulltime field as false for all the documents

```
college> db.students.updateMany({},{$set:{fulltime:false}})
```

```
college> db.students.updateMany({},{$set:{fulltime:false}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
college> db.students.find()
[
  { _id: 1, name: 'person1', age: 20, gpa: 8.9, fulltime: false },
  { _id: 2, name: 'person2', age: 22, gpa: 9, fulltime: false },
  { _id: 3, name: 'person3', age: 25, gpa: 9.5, fulltime: false }
]
```

5. Here in id: 3, we are unsetting the fulltime field

```
college> db.students.updateOne({_id:3},{$unset:{fulltime:''}})
```

```
college> db.students.updateOne({_id:3},{$unset:{fulltime:''}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
college> db.students.find()
[
  { _id: 1, name: 'person1', age: 20, gpa: 8.9, fulltime: false },
  { _id: 2, name: 'person2', age: 22, gpa: 9, fulltime: false },
  { _id: 3, name: 'person3', age: 25, gpa: 9.5 }
]
```

6. Here we use \$exists:false operator to check if value exists or not, wherever the value is not found it sets fulltime to true using \$set operator

```
college> db.students.updateMany({fulltime:{ $exists:false }},{ $set:{fulltime:true}})
```

```
college> db.students.updateMany({fulltime:{ $exists:false }},{ $set:{fulltime:true}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
college> db.students.find()
[
  { _id: 1, name: 'person1', age: 20, gpa: 8.9, fulltime: false },
  { _id: 2, name: 'person2', age: 22, gpa: 9, fulltime: false },
  { _id: 3, name: 'person3', age: 25, gpa: 9.5, fulltime: true }
]
```

Result:

The MongoDB Script to perform update operations is successfully implemented.

Ex.No : 5 Write a MongoDB Script to Update Documents with Aggregation Pipeline

Date :

Aim

To update documents with aggregation pipeline

Algorithm

Step 1: Use insertMany() to add multiple student documents

Step 2: Start an aggregation pipeline on the students collection.

Step 3: Use \$addFields to create a new field called totalScore by adding the math and science scores for each student.

Step 4: Use \$set to add a new field called averageScore.

Step 5: Use \$project to include only the name, totalScore, and averageScore in the output documents.

Step 6: Execute the aggregation pipeline to update and view student documents with calculated scores.

Script to Execute:

```
db.students.insertMany([
{
_id: 1,
name: 'Alice',
maths: 85,
science: 90,
address: 'Kingfield'
},
```

```
{
  _id: 2,
  name: 'Bob',
  maths: 75,
  science: 80,
  address: 'Levay'
},
{
  _id: 3,
  name: 'Charlie',
  maths: 95,
  science: 85,
  address: 'Borovil'
}
])
```

```
db.students.aggregate([
  {
    $addFields: {
      totalScore: { $add: ["$maths", "$science"] }
    }
  },
  {
```



```
$set: {  
  averageScore: { $divide: ["$totalScore", 2] }  
}  
},  
{  
  $project: {  
    name: 1,  
    totalScore: 1,  
    averageScore: 1  
  }  
}  
})
```

Output:

```
record> db.students.insertMany([
...   {
...     _id: 1,
...     name: 'Alice',
...     maths: 85,
...     science: 90,
...     address: 'Kingfield'
...   },
...   {
...     _id: 2,
...     name: 'Bob',
...     maths: 75,
...     science: 80,
...     address: 'Levay'
...   },
...   {
...     _id: 3,
...     name: 'Charlie',
...     maths: 95,
...     science: 85,
...     address: 'Borovil'
...   }
... ])
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3 } }
```

```

record> db.students.aggregate([
...   {
...     $addFields: {
...       totalScore: { $add: ["$maths", "$science"] }
...     }
...   },
...   {
...     $set: {
...       averageScore: { $divide: ["$totalScore", 2] }
...     }
...   },
...   {
...     $project: {
...       name: 1,
...       totalScore: 1,
...       averageScore: 1
...     }
...   }
... ])
[
  { _id: 1, name: 'Alice', totalScore: 175, averageScore: 87.5 },
  { _id: 2, name: 'Bob', totalScore: 155, averageScore: 77.5 },
  { _id: 3, name: 'Charlie', totalScore: 180, averageScore: 90 }
]

```

Result:

The MongoDB Script to perform updates using the aggregation pipeline is successfully executed.

Ex.No : 6 Write a MongoDB Script to Delete Single and Multiple Documents**Date :****Aim**

To write a MongoDB Script to delete single and multiple documents.

Algorithm

Step1: Use insertMany to add six employee documents to the employees collection with fields like _id, name, age, and designation.

Step 2: Call deleteOne to remove one employee document with the designation "Manager".

Step 3: Execute find() to display the current state of the employees collection after the deletion.

Step 4: Use deleteMany to remove all employee documents with the designation "Programmer".

Step 5: Execute find() again to show the final state of the employees collection after the bulk deletion.

Script to Execute:

```
db.employees.insertMany([
{ _id: 1, name: "Kumar", age: 35, designation: "Manager" },
{ _id: 2, name: "Ram", age: 35, designation: "Manager" },
{ _id: 3, name: "Stephen", age: 25, designation: "Programmer" },
{ _id: 4, name: "Subbu", age: 35, designation: "Programmer" },
{ _id: 5, name: "Anbu", age: 22, designation: "Programmer" },
{ _id: 6, name: "Veena", age: 22, designation: "Manager" }
])

//Deletes one document with the designation as manager
```

```
db.employees.deleteOne({designation:"Manager"})
```

```
//To view the output after deleteone
```

```
db.employees.find()
```

```
//Deletes all the documents with designation as programmer
```

```
db.employees.deleteMany({designation:"Programmer"})
```

```
//To view the output after deleteMany
```

```
db.employees.find()
```

Output:

```
record> db.employees.insertMany([
... { _id: 1, name: "Kumar", age: 35, designation: "Manager" },
... { _id: 2, name: "Ram", age: 35, designation: "Manager" },
... { _id: 3, name: "Stephen", age: 25, designation: "Programmer" },
... { _id: 4, name: "Subbu", age: 35, designation: "Programmer" },
... { _id: 5, name: "Anbu", age: 22, designation: "Programmer" },
... { _id: 6, name: "Veena", age: 22, designation: "Manager" }
... ])
{
  acknowledged: true,
  insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4, '4': 5, '5': 6 }
}
```

```
record> db.employees.deleteOne({designation:"Manager"})
{ acknowledged: true, deletedCount: 1 }
record> db.employees.find()
[
  { _id: 2, name: 'Ram', age: 35, designation: 'Manager' },
  { _id: 3, name: 'Stephen', age: 25, designation: 'Programmer' },
  { _id: 4, name: 'Subbu', age: 35, designation: 'Programmer' },
  { _id: 5, name: 'Anbu', age: 22, designation: 'Programmer' },
  { _id: 6, name: 'Veena', age: 22, designation: 'Manager' }
]
```

```
record> db.employees.deleteMany({designation:"Programmer"})
{ acknowledged: true, deletedCount: 3 }
record>

record> //To view the output after deleteMany

record> db.employees.find()
[
  { _id: 2, name: 'Ram', age: 35, designation: 'Manager' },
  { _id: 6, name: 'Veena', age: 22, designation: 'Manager' }
]
```

Result:

The MongoDB Script to delete single and multiple documents is successfully executed.

Ex.No : 7 Write a MongoDB Script to Perform String Aggregation Operations

Date :

Aim

To write a MongoDB Script to perform string aggregation operations.

Algorithm

Step1: Select the database.

Step 2: Create the collection (if it doesn't exist).

Step 3: Insert multiple documents into the collection using insertMany.

Step 4: Perform aggregation on the collection using the aggregate method.

Step 5: Use the \$project stage to apply various transformations (e.g., \$concat, \$split, \$strcasecmp, \$substr, \$toLower, \$toUpperCase).

Step 6: View the aggregated results to verify the output.

Script to Execute:

//Insert Documents in Database

```
db.people.insertMany([
  { "_id": 1, "fullName": "Alice Johnson", "age": 25 },
  { "_id": 2, "fullName": "Bob Smith", "age": 30 },
  { "_id": 3, "fullName": "Charlie Brown", "age": 35 }
])
```

//View the inserted documents

```
db.people.find()
```

//String Aggregation

```
db.people.aggregate([
  {
    $project: {
      _id: 0,
      fullName: 1,
```

```

//Concatenate a String
greeting: { $concat: ["Hello, ", "$fullName", "!"] },

// Split Full Name
nameParts: { $split: ["$fullName", " "] },

// Compare FullName with Alice Johnson
nameComparison: { $strcasecmp: ["$fullName", "ALICE JOHNSON"] },

// Create a substring with first 5 letters
nameSubstring: { $substr: ["$fullName", 0, 5] },

// Print Lower Case
lowerCaseName: { $toLower: "$fullName" },

// Print Upper Case
upperCaseName: { $toUpper: "$fullName" }
}
}
)

```

Output:

```

record> db.people.insertMany([ { "_id": 1, "fullName": "Alice Johnson", "age": 25 }, { "_id": 2, "fullName": "Bob Smith", "age": 30 }, { "_id": 3, "fullName": "Charlie Brown", "age": 35 } ] )
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3 } }

```

```

record> db.people.find()
[
  { _id: 1, fullName: 'Alice Johnson', age: 25 },
  { _id: 2, fullName: 'Bob Smith', age: 30 },
  { _id: 3, fullName: 'Charlie Brown', age: 35 }
]

```



```
record> db.people.aggregate([
...   {
...     $project: {
...       _id: 0,
...       fullName: 1,
...       greeting: {
...         $concat: ["Hello, ", "$fullName", "!"]
...       },
...       nameParts: {
...         $split: ["$fullName", " "]
...       },
...       nameComparison: {
...         $strcasecmp: ["$fullName", "ALICE JOHNSON"]
...       },
...       nameSubstring: {
...         $substr: ["$fullName", 0, 5]
...       },
...       lowerCaseName: {
...         $toLower: "$fullName"
...       },
...       upperCaseName: {
...         $toUpper: "$fullName"
...       }
...     }
...   }
... ])
```

```
[
  {
    fullName: 'Alice Johnson',
    greeting: 'Hello, Alice Johnson!',
    nameParts: [ 'Alice', 'Johnson' ],
    nameComparison: 0,
    nameSubstring: 'Alice',
    lowerCaseName: 'alice johnson',
    upperCaseName: 'ALICE JOHNSON'
  },
  {
    fullName: 'Bob Smith',
    greeting: 'Hello, Bob Smith!',
    nameParts: [ 'Bob', 'Smith' ],
    nameComparison: 1,
    nameSubstring: 'Bob S',
    lowerCaseName: 'bob smith',
    upperCaseName: 'BOB SMITH'
  },
  {
    fullName: 'Charlie Brown',
    greeting: 'Hello, Charlie Brown!',
    nameParts: [ 'Charlie', 'Brown' ],
    nameComparison: 1,
    nameSubstring: 'Charl',
    lowerCaseName: 'charlie brown',
    upperCaseName: 'CHARLIE BROWN'
  }
]
```

Result:

The MongoDB script to perform string aggregation operation is successfully executed.

Ex.No : 8 Design a Data Model for MongoDB using DbVisualizer

Date :

Aim

To design a data model for MongoDB using DbVisualizer

Algorithm

Step 1: Create a user account to connect with DBvisualizer

- Open mongodb shell
- Choose admin database
 - use admin
- Create user (Note: set any username and password of your choice)

```
db.createUser({  
  user: "myuser",  
  pwd: "mypassword",  
  roles: [{ role: "readWrite", db: "mydatabase" }]  
})
```

Step 2 : Open the DBVisualizer & Click to add new database connection

Step 3 : Choose the mongodb driver from the dropdown menu

Step 4 : Enter the the user name & password that was created in **step 1** to connect with mongoDB server

Step 5 : Once the connection is established successfully, open the New Sql Commander from the top menu in the DBVisualizer & execute the mongoBD commands

Scripts Execution in DBVisualizer (Program)

1.Create Database

use shop

2.Create collections

```
db.createCollection("users")  
db.createCollection("products")  
db.createCollection("orders")
```

3.Insert Values into the Collections

```
db.users.insertOne({
  name: "John Doe",
  email: "john.doe@example.com",
  address: {
    street: "123 Main St",
    city: "Anytown",
    state: "CA",
    zip: "12345"
  }
});
```

```
db.products.insertOne({
  name: "mobile",
  brand: "apple",
  specification: {
    color: "green",
    size: "7 inch",
  }
});
```

```
db.orders.insertOne({
  orderid: "123",
  orderdate: "12 june",
  orderspecification: {
    quantity: "10",
    customername: "karthick",
  }
});
```

```
db.products.insertOne({
  name: "ipod",
  brand: "apple",
  specification: {
    color: "blue",
    size: "4 inch",
  }
});
```

The screenshot shows the DbVisualizer interface with a MongoDB connection. The left sidebar displays the database structure, including collections like 'orders' and 'products'. The main editor shows a MongoDB query being executed. The log window at the bottom displays the execution details:

Time	Status	Command	Exec	Fetch	Rows	Message
08:48:43	STARTED					Executing current statement for: 'MongoDB' [MongoDB], Schema
08:48:43	SUCCESS	DB.PRODUCTS.INSERTONE({	0.371			0 OK. No rows were affected
08:48:43	FINISHED		0.371	0	0	Success: 1

The screenshot shows the DbVisualizer interface with the 'products' collection selected. The table view displays the following data:

_id	name	brand	specification
6717196f9911d4517986f9db	mobile	apple	green 7 inch
671719939911d4517986f9dd	ipod	apple	blue 4 inch

The right sidebar shows the 'Text' tab with a 'Select a single grid cell' button. The bottom status bar indicates the evaluation period expires in 17 days.

Result:

The design of Data Model for MongoDB using DbVisualizer is successfully implemented.

Ex.No : 9

Perform CRUD operations using DbVisualizer

Date :

Aim

To perform CRUD operations using DbVisualizer

Algorithm

Step 1: Create a user account to connect with DBvisualizer

- Open mongodb shell
- Choose admin database
 - use admin
- Create user (Note: set any username and password of your choice)

```
db.createUser({
  user: "myuser",
  pwd: "mypassword",
  roles: [{ role: "readWrite", db: "mydatabase" }]
})
```

Step 2 : Open the DBVisualizer & Click to add new database connection

Step 3 : Choose the mongodb driver from the dropdown menu

Step 4 : Enter the the user name & password that was created in **step 1** to connect with mongoDB server

Step 5 : Once the connection is established successfully, open the New Sql Commander from the top menu in the DBVisualizer & execute the mongoBD commands

Scripts Execution in DBVisualizer (Program)

1.Create Database

use shop

2.Create collections

```
db.createCollection("users")
db.createCollection("products")
db.createCollection("orders")
```

3.Insert Values into the Collections

```
db.users.insertOne({
  name: "John Doe",
  email: "john.doe@example.com",
  address: {
    street: "123 Main St",
    city: "Anytown",
    state: "CA",
    zip: "12345"
  }
});
```

```
db.products.insertOne({
  name: "mobile",
  brand: "apple",
  specification: {
    color: "green",
    size: "7 inch",
  }
});
```

```
db.orders.insertOne({
  orderid: "123",
  orderdate: "12 june",
  orderspecification: {
    quantity: "10",
    customername: "karthick",
  }
});
```

```
db.products.insertOne({
  name: "ipod",
  brand: "apple",
  specification: {
    color: "blue",
    size: "4 inch",
  }
});
```

Table: products				
MongoDB/Databases/shop/Collections/products				
Fields	Data	Document Count	Object Id	
<input type="text"/> <input type="button" value="t"/> <input type="button" value="≡"/> <input type="button" value="≡"/> <input type="button" value="v"/> <input type="button" value="<"/> <input type="button" value=">"/>				
_id	name	brand	specification	
			color	size
6717196f9911d4517986f9db	mobile	apple	green	7 inch
671719939911d4517986f9dd	ipod	apple	blue	4 inch

Table: orders				
MongoDB/Databases/shop/Collections/orders				
Fields	Data	Document Count	Object Id	
<input type="text"/> <input type="button" value="t"/> <input type="button" value="≡"/> <input type="button" value="≡"/> <input type="button" value="v"/> <input type="button" value="<"/> <input type="button" value=">"/>				
_id	orderid	orderdate	orderspecification	
			quantity	customername
6717197c9911d4517986f9dc	123	12 june	10	karthick

4.Read Operation

```
db.products.find();
```

46 / 1 [652] INS				
CRLF Transactions not supp				
Log 1: db.products.find() [2] ×				
<input type="text"/> <input type="button" value="t"/> <input type="button" value="≡"/> <input type="button" value="≡"/> <input type="button" value="v"/> <input type="button" value="<"/> <input type="button" value=">"/>				
_id	name	brand	specification	
			color	size
6717196f9911d4517986f9db	mobile	apple	green	7 inch
671719939911d4517986f9dd	ipod	apple	blue	4 inch

5.Update Operation

```
db.products.updateOne(
  { name: "ipod" },
  { $set: { age: 26 } }
)
```



```
db.products.find();
```

50 / 1 [675] INS CRLF Transactions

Log 1: db.products.updateOne({ name:... [2] X

Q t8 = | t8 v <>

_id	name	brand	specification		age
			color	size	
6717196f9911d4517986f9db	mobile	apple	green	7 inch	
671719939911d4517986f9dd	ipod	apple	blue	4 inch	26

7.Delete Operation

```
db.products.deleteOne({name: "ipod"});
```

```
db.products.find();
```

59 / 13 [818] INS

Log 1: db.products.find() [1] X

Q t8 = | t8 v <>

_id	name	brand	specification	
			color	size
6717196f9911d4517986f9db	mobile	apple	green	7 inch

Result:

The CRUD operations using DbVisualizer is successfully implemented.

Ex.No : 10 Create a Zoho CRM account and organize your Tasks, Meetings and Deals

Date :

Aim

To create a Zoho CRM account and organize Tasks, Meetings and Deals

Algorithm

Step 1: Visit Zoho CRM Website

Step 2: Create a Zoho CRM Account:

- Click on the "Sign Up" button.
- Fill in the required details (name, email, etc.) and complete the registration Process.

Step 3: Login to Zoho CRM:

- Once registered, log in to your Zoho CRM account using your credentials.

Step 4: Set Up Your Organization:

- Define your organization's details in the CRM, such as the company name and other relevant information.

Step 5: Organize Tasks:

- From the dashboard, go to the "Tasks" module.
- Click on "Create Task" and enter the details (title, description, due date, etc.).
- Assign the task to team members or yourself and set reminders if needed.

Step 6: Organize Meetings:

- In the CRM dashboard, go to the "Meetings" module (or "Events").
- Schedule a new meeting by entering the meeting title, time, location, and invitees.
- Add reminders and attach relevant documents or links if necessary.

Step 7: Organize Deals:

- Go to the "Deals" module.
- Click "Add New Deal" and fill in the deal details (deal name, amount, expected closing date, etc.).
- Assign the deal to a team member and associate it with the relevant contacts or accounts.

Output:

CRMHomeLeadsContactsAccountsDealsMeetingsTasksCallsReportsAnalyticsProductsServicesProjectsEnterprise-TrialUPGRADE

Create TaskEdit Page LayoutCancelSave and NewSave

Task Information

Task Owner

harini

Subject

Call

Due Date

22/10/2024

Contact

Kris Marrier (Sample)

Account

King (Sample)

Status

Not Started

Priority

High

Reminder

Repeat

Description Information

CRMHomeLeadsContactsAccountsDealsMeetingsTasksCallsReportsAnalyticsProductsServicesProjectsEnterprise-TrialUPGRADE

CallEdit

Hide Details

Task Information

Task Owner

harini

Subject

Call

Due Date

22/10/2024

Contact

Kris Marrier (Sample)

Account

King (Sample)

Status

Not Started

Priority

High

Created By

harini Tue, 22 Oct 2024 03:16 PM

Modified By

harini Tue, 22 Oct 2024 03:16 PM

Reminder

—

Repeat

—

Kris Marrier (Sample)

Info

Timeline

Conversations

Kris Marrier (Sample)

Contact

Send Email

King (Sample)

555-555-5555

555-555-5555

krismarrier@noemail.invalid

0 Followers

More Info

Deal Summary

+ New

\$*

King - \$ 60,000.00

Identify Decision Makers - Oct 24

Open Activities

+ New

Tasks

4

Meetings

0

CRMHomeLeadsContactsAccountsDealsTasksMeetingsCallsReportsAnalyticsProductsServices

Enterprise-Trial
UPGRADE

All Tasks

Create Task

Actions

Tasks by Status

Sort ByNoneAsc

Filter Tasks by

Search

System Defined Filters

Touched Records

Untouched Records

Record Action

Related Records Action

Locked

Filter By Fields

Closed Time

Contact Name

Created By

Created Time

Due Date

Modified By

Modified Time

Not Started5

Deferred0

In Progress8

Completed2

Waiting

Call

22/10/2024

High

harini

Kris Marrier (Sample)

King (Sample)

Email

22/10/2024

High

harini

Kris Marrier (Sample)

mit

Register for upcoming CRM Webin...

21/10/2024

Low

harini

Task Deleted (Sample)

No Tasks found.

Meeting

21/10/2024

Highest

harini

p

mit

Refer CRM Videos

23/10/2024

Normal

harini

Mitsue Tollner (Sample)

Morlong Associates

Get Approval from Manager

22/10/2024

Normal

harini

Task Deleted (Sample)

Complete CRM Getting Started steps

21/10/2024

Highest

harini

John Butt (Sample)

Benton

Complete CRM Getting Started steps

24/10/2024

Normal

harini

Theola Frey (Sample)

Meeting Information

meeting zoon

Location

☐ Make this an online meeting ?

All day☐

From

22/10/2024

04:00 PM

To

22/10/2024

05:00 PM

Host

harini

Participants

3 Selected + Add

Add more details

Cancel

Save

CRMHomeLeadsContactsAccountsDealsMeetingsTasksCallsReportsAnalyticsProductsServicesProjects...Enterprise-TrialUPGRADE

Meeting created successfully. View details

Create MeetingActions

Total Records 1210 Records Per Page1-10

Filter Meetings by

Search

▼ System Defined Filters

☐ Touched Records

☐ Untouched Records

☐ Record Action

☐ Related Records Action

▼ Filter By Fields

☐ All day

☐ Check-In Address

☐ Check-In By

☐ Check-In City

☐ Check-In Country

☐ Check-In State

☐ Check-In Sub-Locality

☐ Check-In Time

☐ Checked In Status

<input type="checkbox"/>	Title	All	From	To	Related To	Contact Name	Host
<input type="checkbox"/>	meeting zoon		22/10/2024 04:00 PM	22/10/2024 05:00 PM			harini
<input type="checkbox"/>	harini meeting		22/10/2024 04:00 PM	22/10/2024 05:00 PM			harini
<input type="checkbox"/>	New Meeting		21/10/2024 01:00 PM	21/10/2024 02:00 PM			harini
<input type="checkbox"/>	Demo		21/10/2024 03:00 PM	21/10/2024 04:00 PM	Printing Dimensions	Donette Foller (Sample)	harini
<input type="checkbox"/>	Webinar		21/10/2024 05:00 PM	21/10/2024 06:00 PM	Commercial Press (Sample)	Leota Dillard (Sample)	harini
<input type="checkbox"/>	TradeShow		21/10/2024	21/10/2024	Chemel	James Venere (Sample)	harini
<input type="checkbox"/>	Webinar		21/10/2024 04:00 PM	21/10/2024 07:00 PM	Chanay (Sample)	Josephine Darakjiy (Sample)	harini
<input type="checkbox"/>	Seminar		21/10/2024 03:00 PM	21/10/2024 05:00 PM	@ Carissa Kidman (Sample)		harini
<input type="checkbox"/>	Attend Customer conference		21/10/2024	21/10/2024	Feltz Printing Service	Capla Paprocki (Sample)	harini
<input type="checkbox"/>	CRM Webinar		21/10/2024 02:00 PM	21/10/2024 04:00 PM	Morlong Associates	Mitsue Tollner (Sample)	harini

CRMHomeLeadsContactsAccountsDealsMeetingsTasksCallsReportsAnalyticsProductsServicesProjects...Enterprise-TrialUPGRADE

Create DealEdit Page LayoutCancelSave and NewSave

Deal Information

Deal Owner

harini

Deal Name

leads

Account Name

Morlong Associates (Sample)

Type

-None-

Next Step

Lead Source

-None-

Contact Name

Kris Marrier (Sample)

Amount

\$ 230000

Closing Date

22/10/2024

Stage

Qualification

Probability (%)

10

Expected Revenue

\$ 23000.00

Campaign Source

Description Information

Description

Client Script

leads - \$230,000.00

Send Email Edit ...

Related List

- Notes
- Attachments
- Stage History 1
- Competitors
- Cadences
- Open Activities
- Closed Activities
- Products
- Quotes
- Sales Orders
- Contact Roles 1
- Cases
- Emails
- Invoices

Overview Timeline

START 22/10/2024 CLOSING 22/10/2024

Qualification Needs Analysis Value Proposition Identify Decision Makers Proposal/Price Quote Negotiation/Review Closed Won Closed Lost

Deal Owner	harini
Stage	Qualification
Probability (%)	10
Expected Revenue	\$23,000.00
Closing Date	22/10/2024

Contact Person

Kris Marrier (Sample)

Here is your Smart Chat (Ctrl+Space)

Ask Zia Help

All Deals

STAGEVIEW

Sort By None Asc

Filter Deals by

Search

System Defined Filters

- ☐ Touched Records
- ☐ Untouched Records
- ☐ Record Action
- ☐ Related Records Action
- ☐ Locked
- ☐ Email Sentiment
- ☐ Latest Email Status
- ☐ Activities
- ☐ Notes
- ☐ Cadences

Filter By Fields

- ☐ Account Name
- ☐ Amount

Qualification 3 - 10%	Needs Analysis 2 - 20%	Value Proposition 1 - 40%	Identify Decision Makers 2 - 60%	Proposal 1 - 25%
leads Morlong Associates (Sample) harini Kris Marrier (Sample) \$230,000.00 22/10/2024	miit mit harini p \$3,000,000.00 21/10/2024	Chemel Chemel (Sample) harini James Venere (Sample) \$70,000.00 22/10/2024	King King (Sample) harini Kris Marrier (Sample) \$60,000.00 24/10/2024	Printi Printi (Sample) harini Donet \$25,000.00 27/10/2024
Benton Benton (Sample) harini John Butt (Sample) \$250,000.00 22/10/2024	Truhlar And Truhlar Attys Truhlar And Truhlar (Sample) harini Sage Wieser (Sample) \$45,000.00 22/10/2024		Feltz Printing Service Feltz Printing Service (Sample) harini Capla Paprocki (Sample) \$45,000.00 25/10/2024	

Result:

The Zoho CRM account was created and the organization of Tasks, Meeting and Deals is successfully implemented.

Ex.No : 11 Create and Maintain a Project using Zoho CRM Features

Date :

Aim

To create and maintain a project using Zoho CRM feature.

Algorithm

Step 1: Log in to Zoho CRM

Step 2: Navigate to Projects Module

Step 3: Create a New Project

- Click on the Create New button .
- Fill in the required details such as:
 - Project Name
 - Project Description
 - Start Date
 - End Date
 - Owner.

Step 4: Save the Project

Step 5: Add Tasks to the Project

- Navigate to the Tasks section within the project you just created.
- Click on Add Task or New Task.
- Fill in the task details:
 - Task Name.
 - Task Description.
 - Due Date.
 - Priority.
 - Assign To.

Step 6: Save the task for maintaining the project.

Output

New Project

Standard Layout

Project Title * ⓘ

Morlong Associates

Owner

harini

Template

Select

Start Date

End Date

☐ Make this a strict project ⓘ

Description

B I U Puvi 13 A

Zia

Add

Cancel

Browse Templates

Ask Zia

The screenshot shows the Zoho CRM interface. The top navigation bar includes CRM, Home, Leads, Contacts, Accounts, Deals, Meetings, Tasks, Calls, Reports, Analytics, Products, Services, and Projects. The left sidebar contains Home, Feed, Discuss, Reports, Calendar, Projects, Overview, Tasks, Issues, and Recent Projects. The main area displays the 'Tasks' section for 'Morlong Associates'. A table lists tasks with columns: ID, Task Name, Associated..., Owner, Status, Tags, Start Date, Due Date, and Duration. A task with ID MA3-T1 and name 'service based' is shown with status 'Open'. An 'Add Task' button is visible below the table.

ID	Task Name	Associated...	Owner	Status	Tags	Start Date	Due Date	Duration
MA3-T1	service based	Not Associated	Unassigned	Open				

The screenshot shows the 'New Task' form in Zoho CRM. The form includes a 'Task Name' field with the value 'service based'. Below it is an 'Add Description' section with a rich text editor. The 'Task List' dropdown is set to 'General'. There is a file upload area with the text 'Drop files or add attachments here...' and 'Maximum 10 files'. At the bottom, there are 'Add', 'Add More', and 'Cancel' buttons. The Zoho CRM logo and 'Morlong Associates' are visible in the top right corner of the form.

Result

The creation of project and maintenance using Zoho CRM features is successfully implemented.