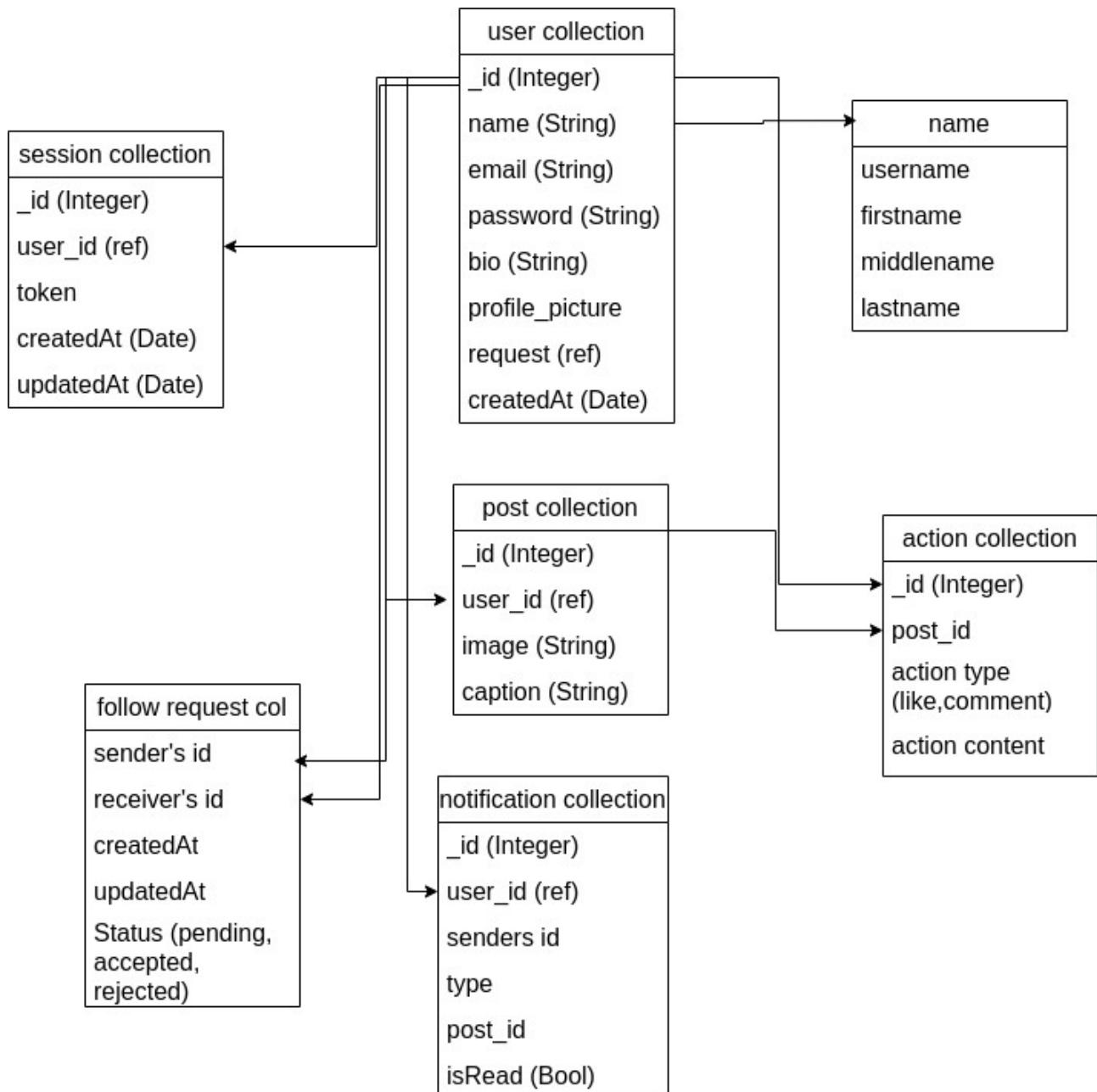
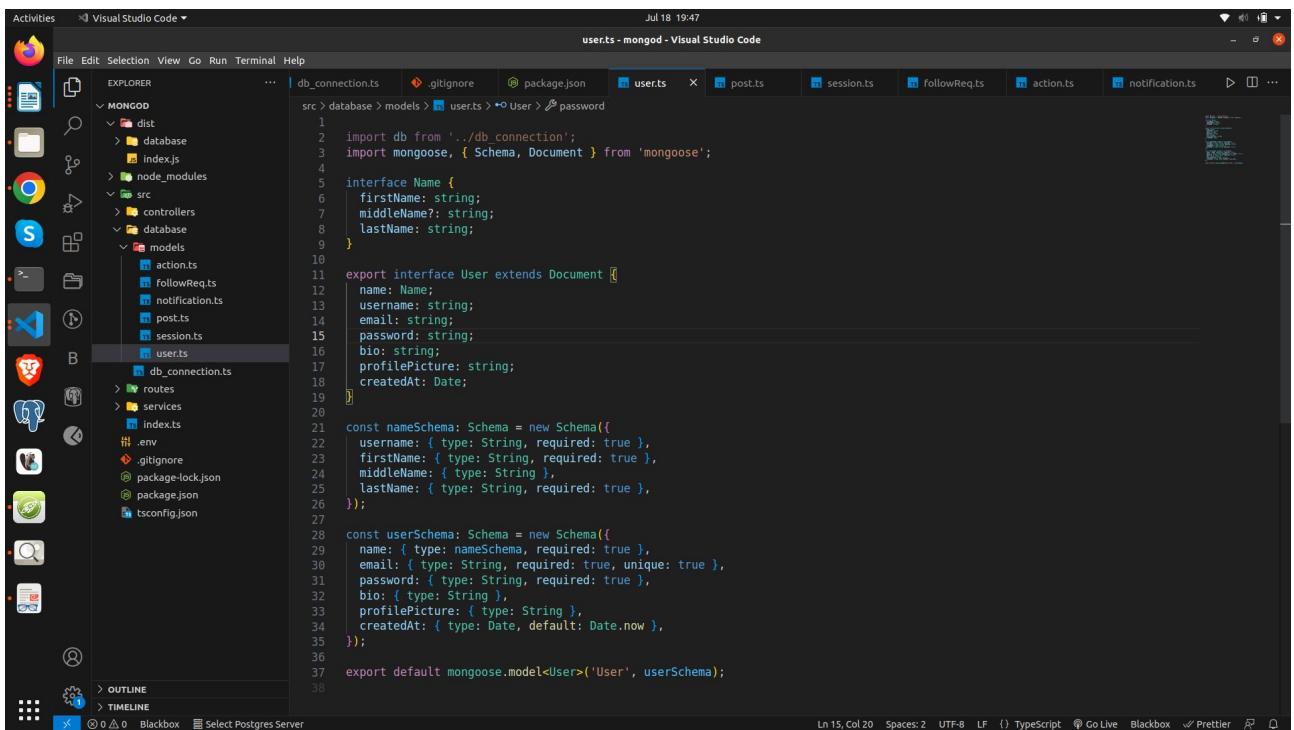


MongoDB Assignment

Schema for Instagram



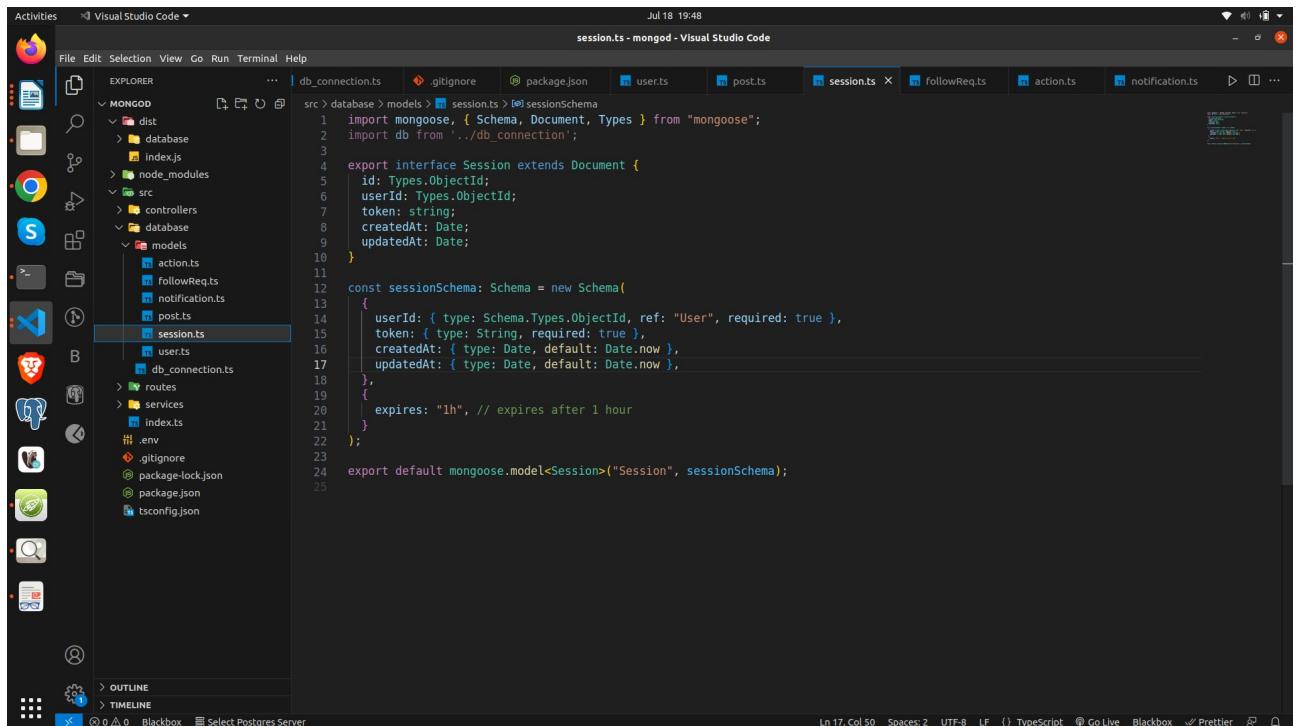
for user Schema Model:



The screenshot shows the Visual Studio Code interface with the file `user.ts` open in the editor. The code defines a `Name` interface and a `User` schema. The `Name` interface has properties for `firstName`, `middleName?`, and `lastName`. The `User` schema extends `Document` and includes properties for `name`, `username`, `email`, `password`, `bio`, `profilePicture`, and `createdAt`. It also defines two schemas: `nameSchema` and `userSchema`, and exports the `User` model.

```
1  import db from '../db.connection';
2  import mongoose, { Schema, Document } from 'mongoose';
3
4  interface Name {
5    firstName: string;
6    middleName?: string;
7    lastName: string;
8  }
9
10 export interface User extends Document {
11   name: Name;
12   username: string;
13   email: string;
14   password: string;
15   bio: string;
16   profilePicture: string;
17   createdAt: Date;
18 }
19
20 const nameSchema: Schema = new Schema({
21   username: { type: String, required: true },
22   firstName: { type: String, required: true },
23   middleName: { type: String },
24   lastName: { type: String, required: true },
25 });
26
27
28 const userSchema: Schema = new Schema({
29   name: { type: nameSchema, required: true },
30   email: { type: String, required: true, unique: true },
31   password: { type: String, required: true },
32   bio: { type: String },
33   profilePicture: { type: String },
34   createdAt: { type: Date, default: Date.now },
35 });
36
37 export default mongoose.model<User>('User', userSchema);
```

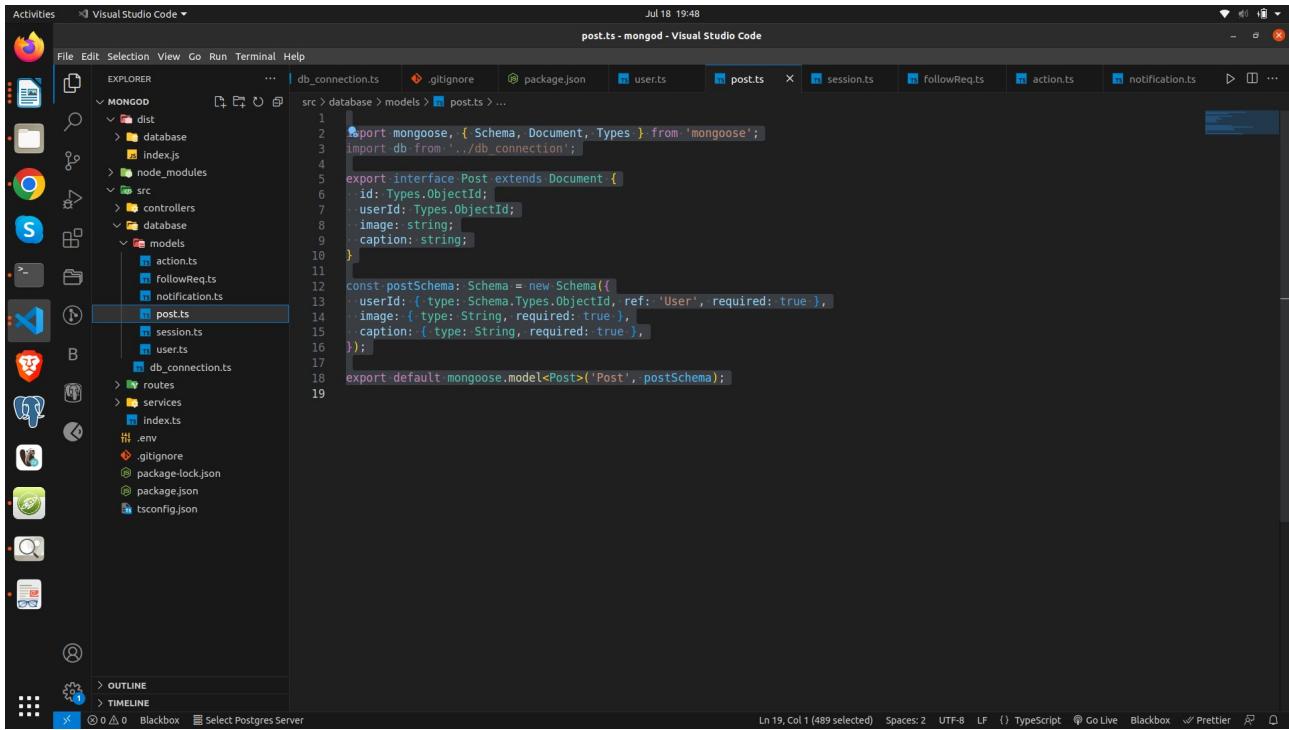
for Session model Schema:



The screenshot shows the Visual Studio Code interface with the file `session.ts` open in the editor. The code defines a `Session` interface and a `sessionSchema` schema. The `Session` interface extends `Document` and includes properties for `id`, `userId`, `token`, `createdAt`, and `updatedAt`. The `sessionSchema` schema extends `sessionSchema` and includes a `expires` field with a value of `'1h'`. It also defines a `user` reference and `createdAt` and `updatedAt` fields. The code exports the `Session` model.

```
1  import mongoose, { Schema, Document, Types } from "mongoose";
2  import db from '../db_connection';
3
4  export interface Session extends Document {
5    id: Types.ObjectId;
6    userId: Types.ObjectId;
7    token: string;
8    createdAt: Date;
9    updatedAt: Date;
10 }
11
12 const sessionSchema: Schema = new Schema({
13   userId: { type: Schema.Types.ObjectId, ref: "User", required: true },
14   token: { type: String, required: true },
15   createdAt: { type: Date, default: Date.now },
16   updatedAt: { type: Date, default: Date.now },
17 },
18 {
19   expires: "1h", // expires after 1 hour
20 }
21
22 );
23
24 export default mongoose.model<Session>("Session", sessionSchema);
```

for posts schema model:

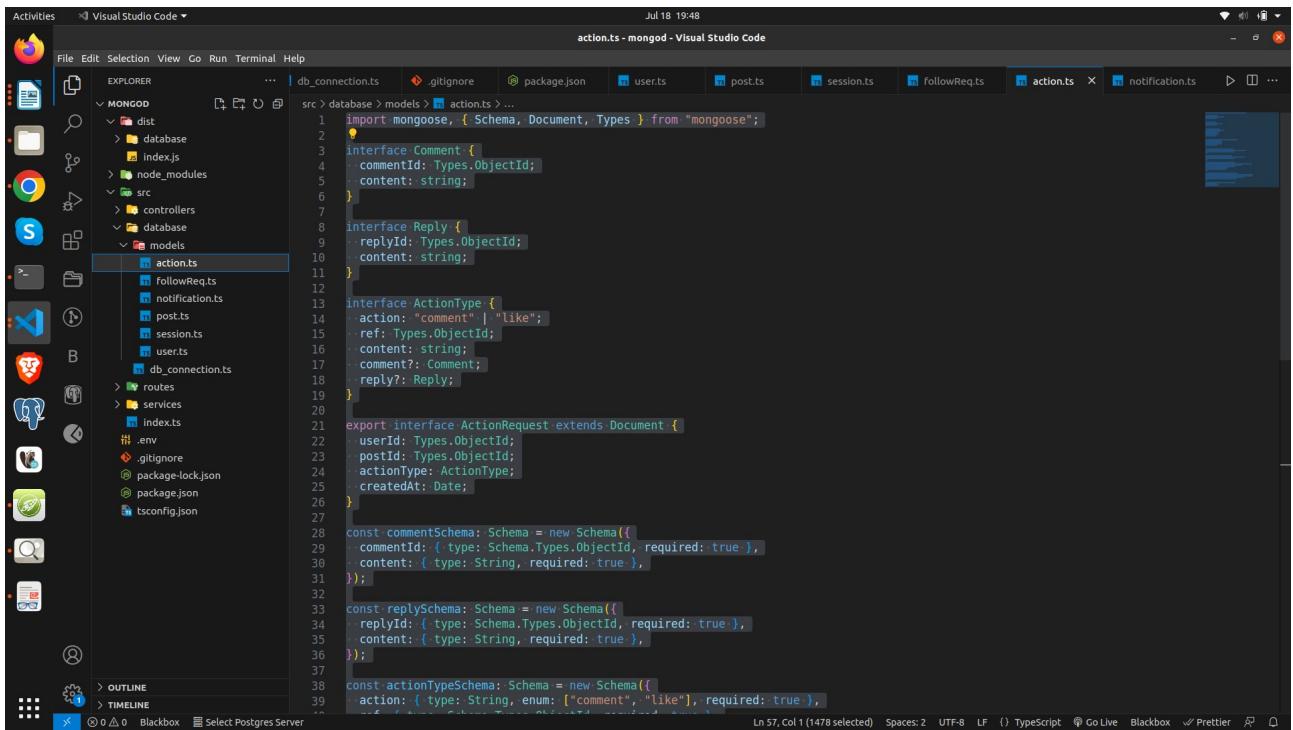


The screenshot shows the Visual Studio Code interface with the 'post.ts' file open in the editor. The file content is as follows:

```
Jul 18 19:48
post.ts - mongod - Visual Studio Code

1 import mongoose, { Schema, Document, Types } from 'mongoose';
2
3 export interface Post extends Document {
4   id: Types.ObjectId;
5   userId: Types.ObjectId;
6   image: string;
7   caption: string;
8 }
9
10 const postSchema: Schema = new Schema({
11   userId: { type: Schema.Types.ObjectId, ref: 'User', required: true },
12   image: { type: String, required: true },
13   caption: { type: String, required: true },
14 });
15
16 export default mongoose.model<Post>('Post', postSchema);
```

for actionResponse model schema:

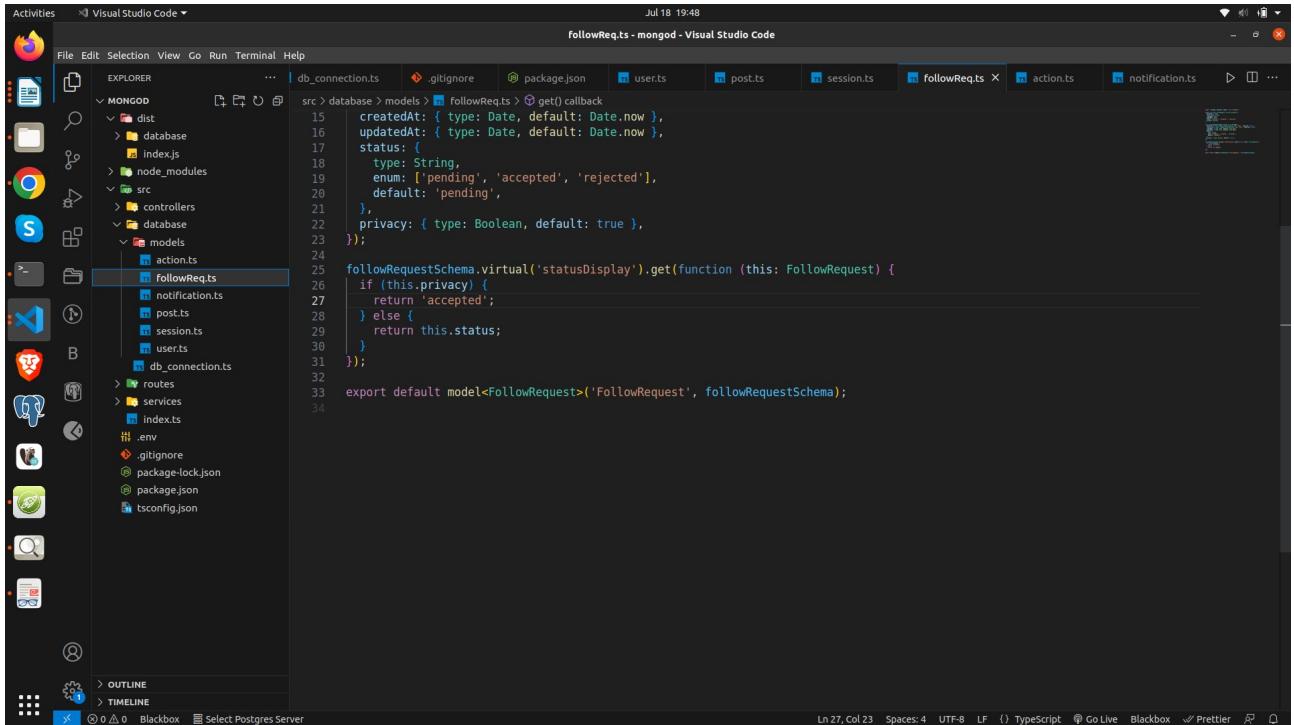


The screenshot shows the Visual Studio Code interface with the 'action.ts' file open in the editor. The file content is as follows:

```
Jul 18 19:48
action.ts - mongod - Visual Studio Code

1 import mongoose, { Schema, Document, Types } from "mongoose";
2
3 interface Comment {
4   commentId: Types.ObjectId;
5   content: string;
6 }
7
8 interface Reply {
9   replyId: Types.ObjectId;
10  content: string;
11 }
12
13 interface ActionType {
14   action: "comment" | "like";
15   ref: Types.ObjectId;
16   content: string;
17   comment?: Comment;
18   reply?: Reply;
19 }
20
21 export interface ActionRequest extends Document {
22   userId: Types.ObjectId;
23   postId: Types.ObjectId;
24   actionPerformed: ActionType;
25   createdAt: Date;
26 }
27
28 const commentSchema: Schema = new Schema({
29   commentId: { type: Schema.Types.ObjectId, required: true },
30   content: { type: String, required: true },
31 });
32
33 const replySchema: Schema = new Schema({
34   replyId: { type: Schema.Types.ObjectId, required: true },
35   content: { type: String, required: true },
36 });
37
38 const actionPerformedSchema: Schema = new Schema({
39   action: { type: String, enum: ["comment", "like"], required: true },
```

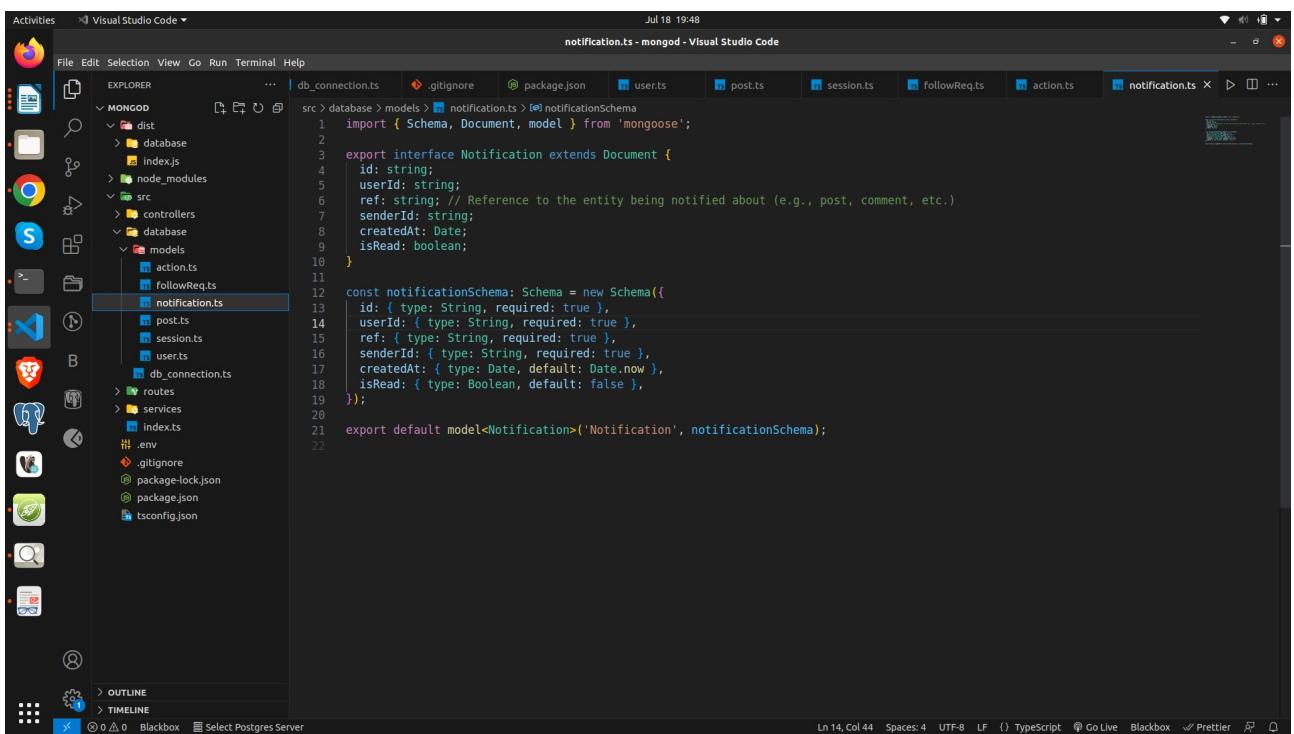
for FollowRequest schema model:



A screenshot of Visual Studio Code showing the code editor with the file `followReq.ts` open. The code defines a schema for a `FollowRequest` document. It includes fields for `createdAt`, `updatedAt`, `status` (enum: pending, accepted, rejected), `privacy` (Boolean), and a virtual method `statusDisplay` that returns 'accepted' if `privacy` is true, otherwise the `status`. The file is part of a larger project structure under the `MONGOD` folder.

```
src > database > models > followReq.ts > get() callback
15  createdAt: { type: Date, default: Date.now },
16  updatedAt: { type: Date, default: Date.now },
17  status: {
18    type: String,
19    enum: ['pending', 'accepted', 'Rejected'],
20    default: 'pending',
21  },
22  privacy: { type: Boolean, default: true },
23}
24
25 followRequestSchema.virtual('statusDisplay').get(function (this: FollowRequest) {
26   if (this.privacy) {
27     return 'accepted';
28   } else {
29     return this.status;
30   }
31 });
32
33 export default model<FollowRequest>('FollowRequest', followRequestSchema);
```

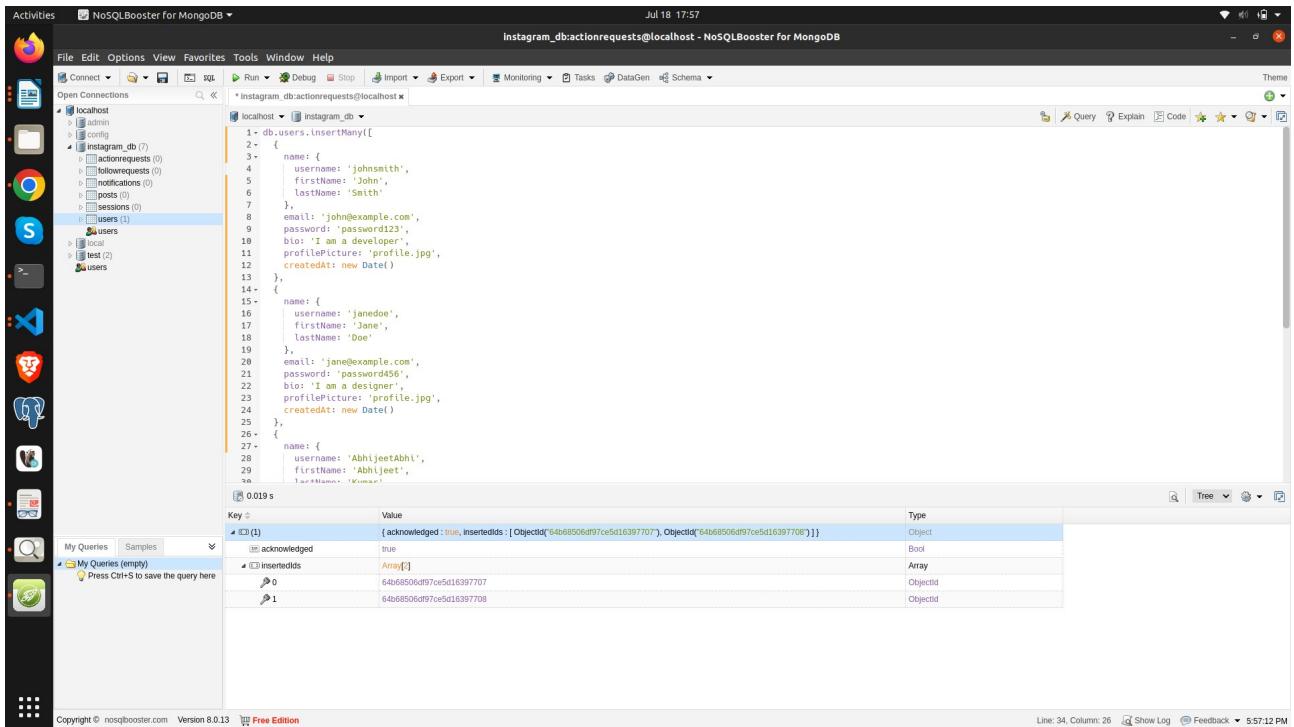
for notification schema model:



A screenshot of Visual Studio Code showing the code editor with the file `notification.ts` open. The code defines a schema for a `Notification` document, extending the `Document` interface. It includes fields for `id`, `userId`, `ref` (a reference to the entity being notified about), `senderId`, `createdAt`, and `isRead`. The file is part of a larger project structure under the `MONGOD` folder.

```
src > database > models > notification.ts > notificationSchema
1  import { Schema, Document, model } from 'mongoose';
2
3  export interface Notification extends Document {
4    id: string;
5    userId: string;
6    ref: string; // Reference to the entity being notified about (e.g., post, comment, etc.)
7    senderId: string;
8    createdAt: Date;
9    isRead: boolean;
10 }
11
12 const notificationSchema: Schema = new Schema({
13   id: { type: String, required: true },
14   userId: { type: String, required: true },
15   ref: { type: String, required: true },
16   senderId: { type: String, required: true },
17   createdAt: { type: Date, default: Date.now },
18   isRead: { type: Boolean, default: false },
19 });
20
21 export default model<Notification>('Notification', notificationSchema);
```

for Insert in user model:



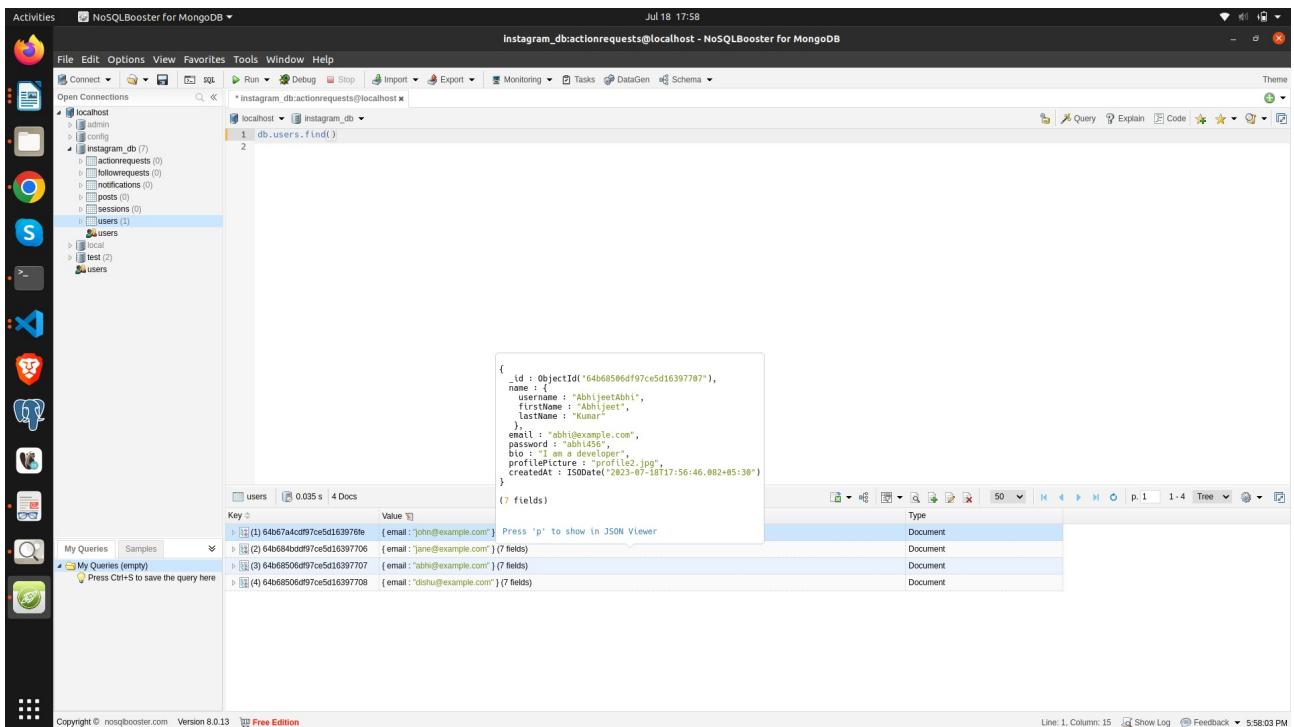
The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar lists databases (localhost, config, Instagram_db) and collections (actionrequests, followrequests, notifications, posts, users). The 'users' collection is selected. The main area shows a query script for inserting multiple users:

```
1- db.users.insertMany([
2-   {
3-     name: {
4-       username: 'johnsmith',
5-       firstName: 'John',
6-       lastName: 'Smith'
7-     },
8-     email: 'john@example.com',
9-     password: 'password123',
10-    bio: 'I am a developer',
11-    profilePicture: 'profile.jpg',
12-    createdAt: new Date()
13-  },
14-  {
15-    name: {
16-      username: 'janedoe',
17-      firstName: 'Jane',
18-      lastName: 'Doe'
19-    },
20-    email: 'janedoe@example.com',
21-    password: 'password456',
22-    bio: 'I am a designer',
23-    profilePicture: 'profile.jpg',
24-    createdAt: new Date()
25-  },
26-  {
27-    name: {
28-      username: 'AbhijeeetAbhi',
29-      firstName: 'Abhijeeet',
30-    }
31-  }
32-])
```

The results table shows the inserted documents:

Key	Value	Type
0 (1)	{ acknowledged: true, insertedIds: [ObjectId("64b68506df97ce5d16397707"), ObjectId("64b68506df97ce5d16397708")] }	Object
0	64b68506df97ce5d16397707	Array
1	64b68506df97ce5d16397708	Objectid

for find Query in user model:



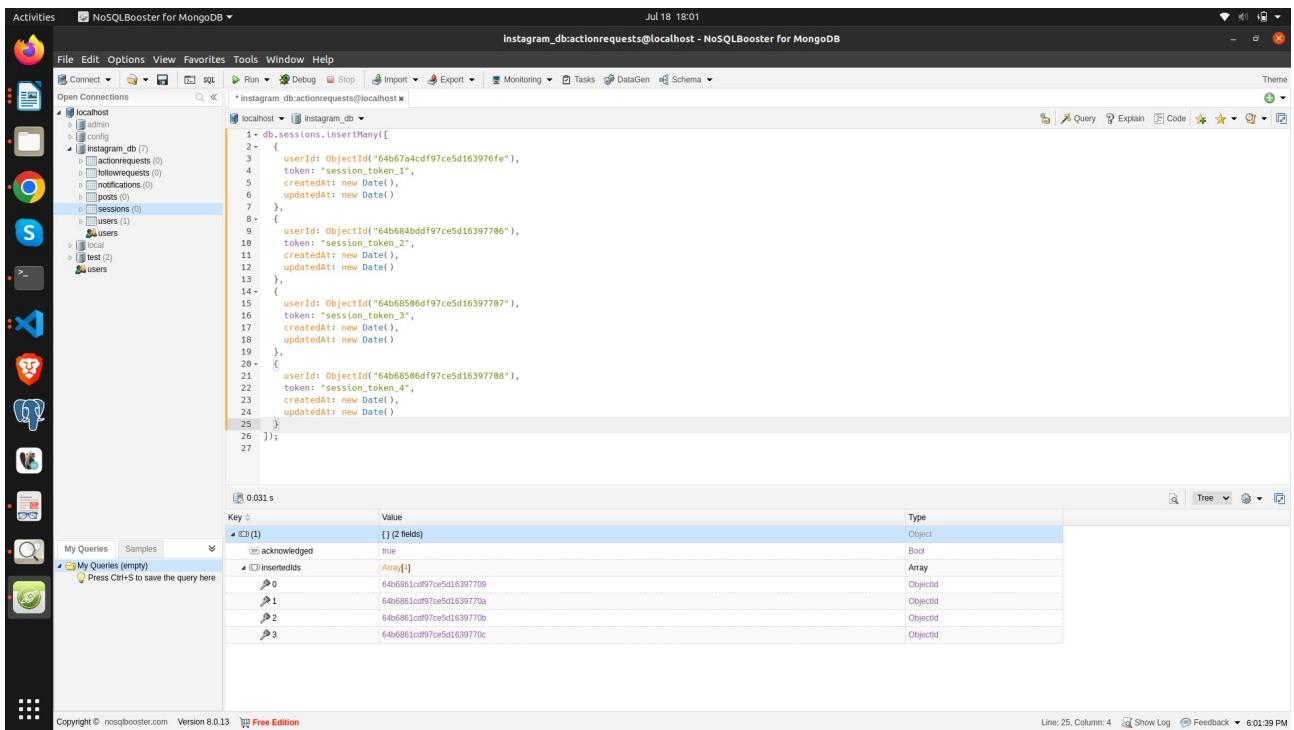
The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar lists databases (localhost, config, Instagram_db) and collections (actionrequests, followrequests, notifications, posts, users). The 'users' collection is selected. The main area shows a query script for finding documents:

```
1- db.users.find()
```

The results table shows the found documents:

Key	Value	Type
0 (1)	{ _id: ObjectId("64b68506df97ce5d16397707"), name: { username: 'AbhijeeetAbhi', firstName: 'Abhijeeet', lastName: 'Abhi' }, email: 'abhi@example.com', password: 'abhi456', bio: 'I am a developer', profilePicture: 'profile.jpg', createdAt: ISODate("2023-07-10T17:56:46.002+05:30") }	Document
1 (2)	{ _id: ObjectId("64b68506df97ce5d16397706"), email: 'john@example.com' }	Document
2 (3)	{ _id: ObjectId("64b68506df97ce5d16397705"), email: 'janedoe@example.com' }	Document
3 (4)	{ _id: ObjectId("64b68506df97ce5d16397708"), email: 'dishes@example.com' }	Document

for inserting in session schema model:



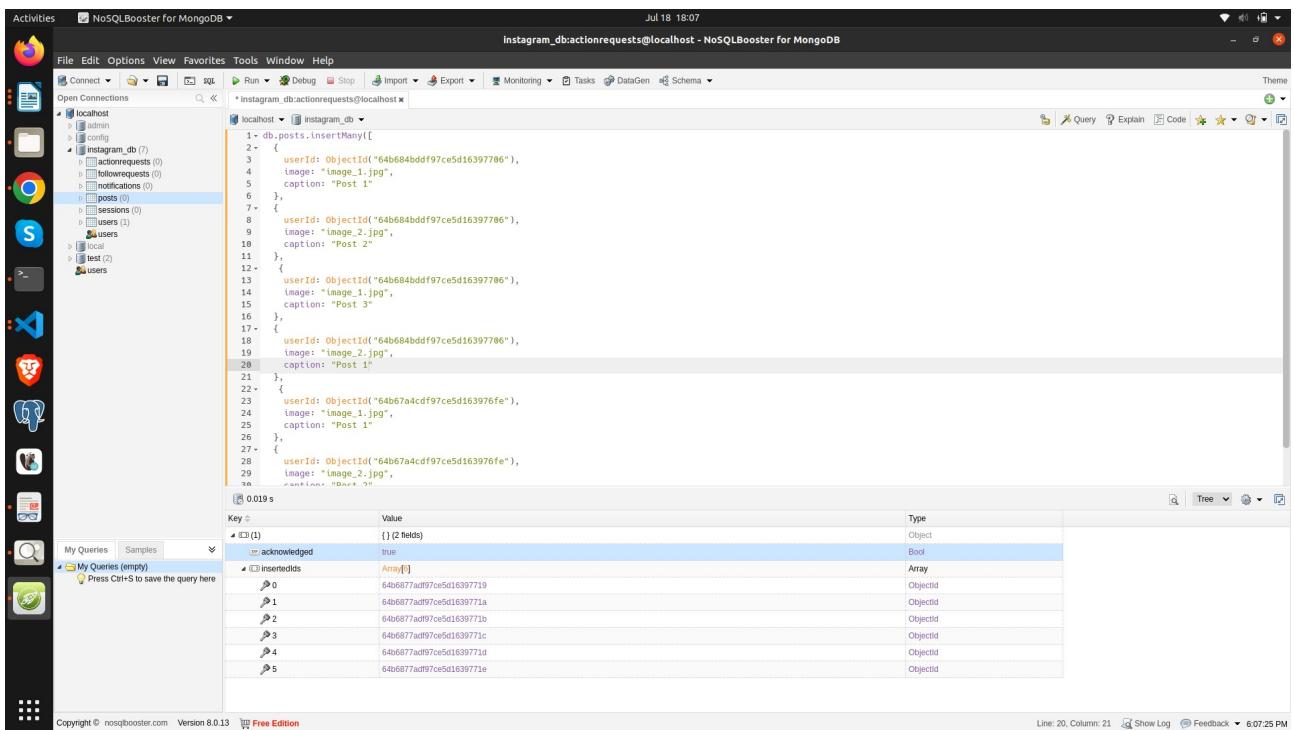
The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar lists databases: admin, config, Instagram_db (with collections: actionrequests, followerrequests, notifications, posts, sessions, users), local, and test (with collections: users). The main area shows a query in the mongo shell. The query is:

```
1 db.sessions.insertMany([
2 {
3   userId: ObjectId("64b67a4cdf97ce5d163976fe"),
4   token: "session_token_1",
5   createdAt: new Date(),
6   updatedAt: new Date()
7 },
8 {
9   userId: ObjectId("64b684bddf97ce5d16397706"),
10 token: "session_token_2",
11 createdAt: new Date(),
12 updatedAt: new Date()
13 },
14 {
15   userId: ObjectId("64b68506df97ce5d16397707"),
16 token: "session_token_3",
17 createdAt: new Date(),
18 updatedAt: new Date()
19 },
20 {
21   userId: ObjectId("64b68506df97ce5d16397708"),
22 token: "session_token_4",
23 createdAt: new Date(),
24 updatedAt: new Date()
25 }
26 ]);
27
```

The results table shows 4 inserted documents with the following details:

Key	Value	Type
0 acknowledged	true	Object
0 insertedIds	[4]	Array
0 insertedIds.0	64b6861cd97ce5d16397709	Objectid
0 insertedIds.1	64b6861cd97ce5d1639770a	Objectid
0 insertedIds.2	64b6861cd97ce5d1639770b	Objectid
0 insertedIds.3	64b6861cd97ce5d1639770c	Objectid

for inserting in session schema model:



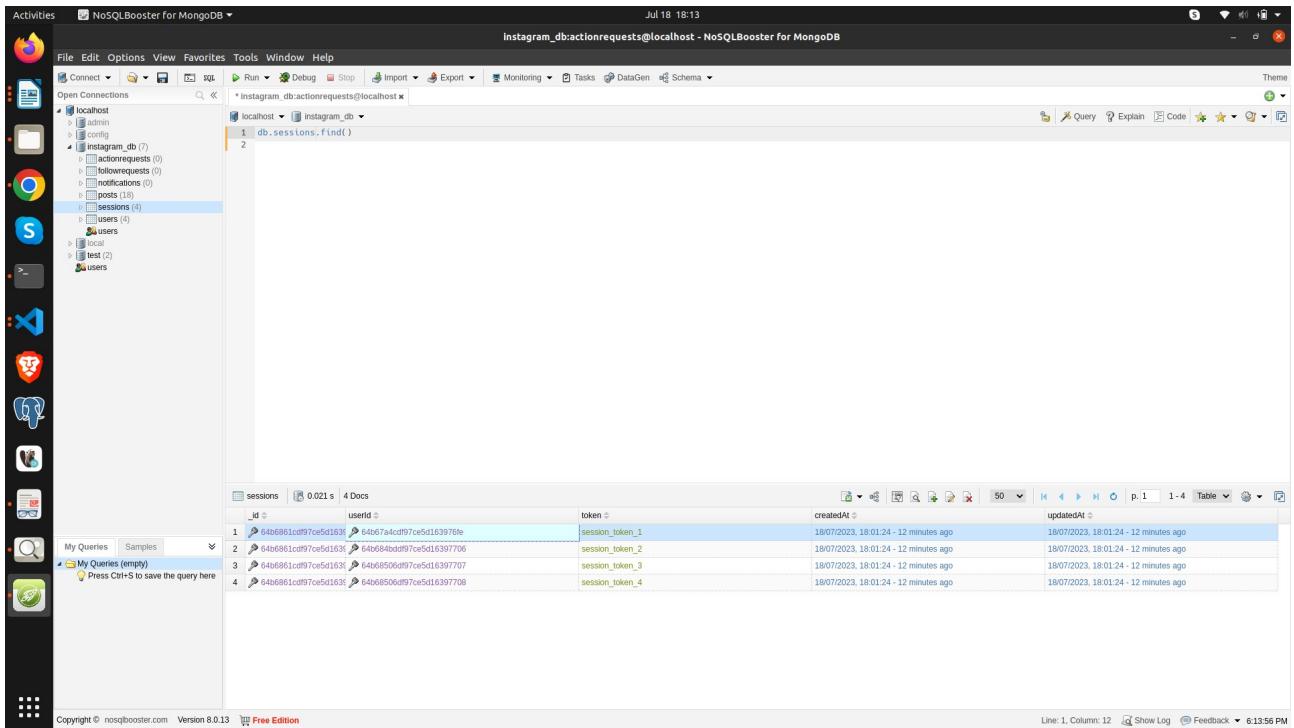
The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar lists databases: admin, config, Instagram_db (with collections: actionrequests, followerrequests, notifications, posts, sessions, users), local, and test (with collections: users). The main area shows a query in the mongo shell. The query is:

```
1 db.posts.insertMany([
2 {
3   userId: ObjectId("64b684bddf97ce5d16397706"),
4   image: "Image_1.jpg",
5   caption: "Post 1"
6 },
7 {
8   userId: ObjectId("64b684bddf97ce5d16397707"),
9   image: "Image_2.jpg",
10 caption: "Post 2"
11 },
12 {
13   userId: ObjectId("64b684bddf97ce5d16397708"),
14   image: "Image_1.jpg",
15   caption: "Post 3"
16 },
17 {
18   userId: ObjectId("64b684bddf97ce5d16397709"),
19   image: "Image_2.jpg",
20   caption: "Post 1"
21 },
22 {
23   userId: ObjectId("64b67a4cdf97ce5d163976fe"),
24   image: "Image_1.jpg",
25   caption: "Post 1"
26 },
27 {
28   userId: ObjectId("64b67a4cdf97ce5d163976fe"),
29   image: "Image_2.jpg",
30   caption: "Post 2"
31 }
32 ]);
33
```

The results table shows 6 inserted documents with the following details:

Key	Value	Type
0 acknowledged	true	Object
0 insertedIds	[6]	Array
0 insertedIds.0	64b6877ad97ce5d16397719	Objectid
0 insertedIds.1	64b6877ad97ce5d1639771a	Objectid
0 insertedIds.2	64b6877ad97ce5d1639771b	Objectid
0 insertedIds.3	64b6877ad97ce5d1639771c	Objectid
0 insertedIds.4	64b6877ad97ce5d1639771d	Objectid
0 insertedIds.5	64b6877ad97ce5d1639771e	Objectid

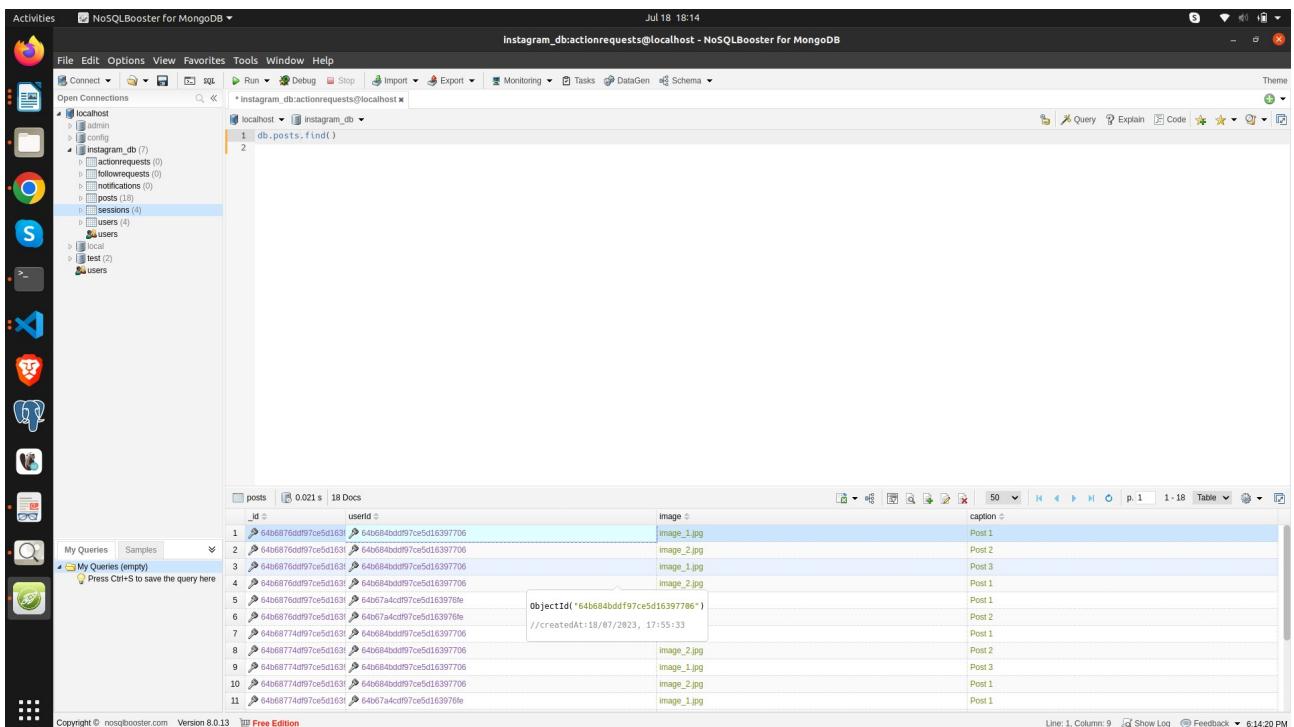
for find Query in sessions model:



The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar lists databases: admin, config, Instagram_db (with actionrequests, followerrequests, notifications, posts, sessions, users), local, and test (2). The 'sessions' collection under Instagram_db is selected. The main pane shows a query builder with the command `db.sessions.find()`. Below the query, a table displays 4 documents from the sessions collection. Each document has fields: _id, userId, token, createdAt, and updatedAt. The table shows 4 rows of data, each with a unique _id and token, and identical createdAt and updatedAt values.

_id	userId	token	createdAt	updatedAt
1	64b6861cd97ce5d163	64b67a4cd97ce5d163976e	18/07/2023, 18:01:24 - 12 minutes ago	18/07/2023, 18:01:24 - 12 minutes ago
2	64b6861cd97ce5d163	64b684bdd97ce5d16397706	18/07/2023, 18:01:24 - 12 minutes ago	18/07/2023, 18:01:24 - 12 minutes ago
3	64b6861cd97ce5d163	64b68506d97ce5d16397707	18/07/2023, 18:01:24 - 12 minutes ago	18/07/2023, 18:01:24 - 12 minutes ago
4	64b6861cd97ce5d163	64b68506d97ce5d16397708	18/07/2023, 18:01:24 - 12 minutes ago	18/07/2023, 18:01:24 - 12 minutes ago

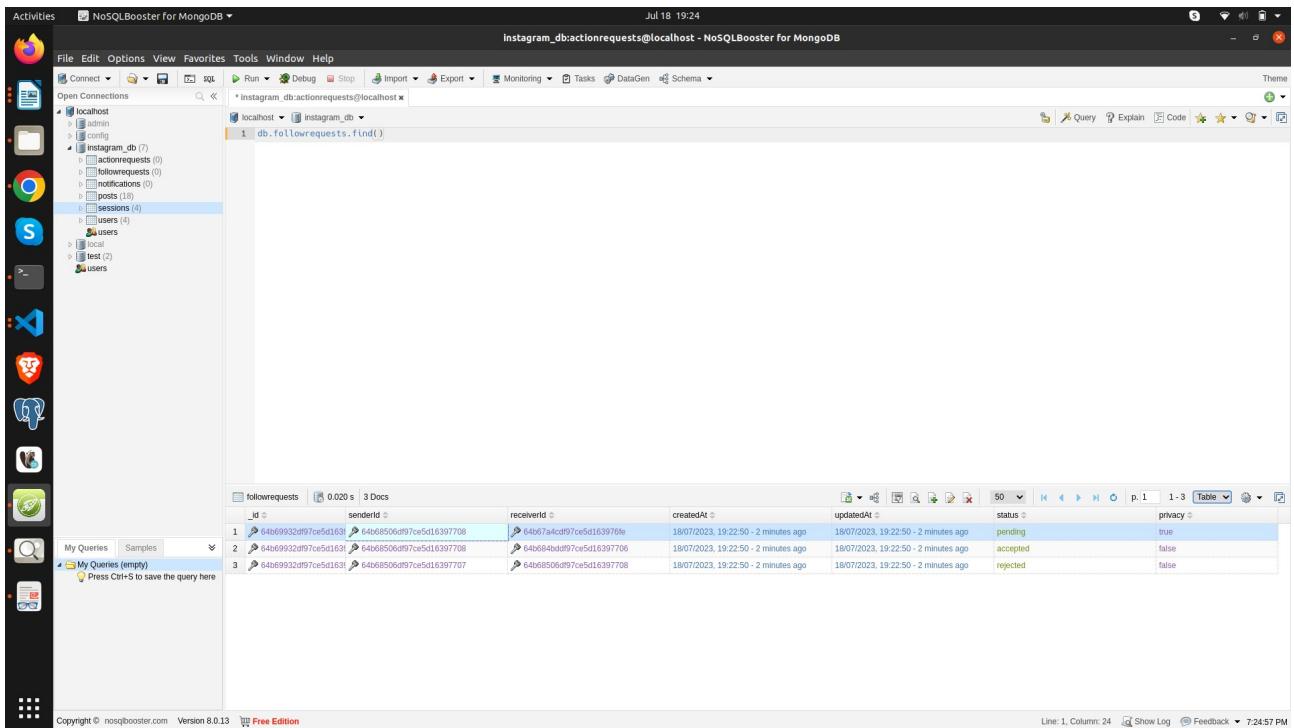
for find Query in posts model:



The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar lists databases: admin, config, Instagram_db (with actionrequests, followerrequests, notifications, posts, sessions, users), local, and test (2). The 'posts' collection under Instagram_db is selected. The main pane shows a query builder with the command `db.posts.find()`. Below the query, a table displays 18 documents from the posts collection. Each document has fields: _id, userId, image, and caption. The table shows 18 rows of data, each with a unique _id and image, and identical caption values.

_id	userId	image	caption
1	64b68760d97ce5d163	image_1.jpg	Post 1
2	64b68760d97ce5d163	image_2.jpg	Post 2
3	64b68760d97ce5d163	image_1.jpg	Post 3
4	64b68760d97ce5d163	image_2.jpg	Post 1
5	64b68760d97ce5d163	image_1.jpg	Post 1
6	64b68760d97ce5d163	image_2.jpg	Post 2
7	64b68774d97ce5d163	image_1.jpg	Post 1
8	64b68774d97ce5d163	image_2.jpg	Post 2
9	64b68774d97ce5d163	image_1.jpg	Post 3
10	64b68774d97ce5d163	image_2.jpg	Post 1
11	64b68774d97ce5d163	image_1.jpg	Post 1

for find Query in followRequest model:



Activities NoSQLBooster for MongoDB Jul 18 19:24

File Edit Options View Favorites Tools Window Help

localhost Instagram_db (7) Instagram_db (7) sessions (4) users (4) posts (18) notifications (0) followrequests (0) admin config

localhost Instagram_db 1 db.followrequests.find()

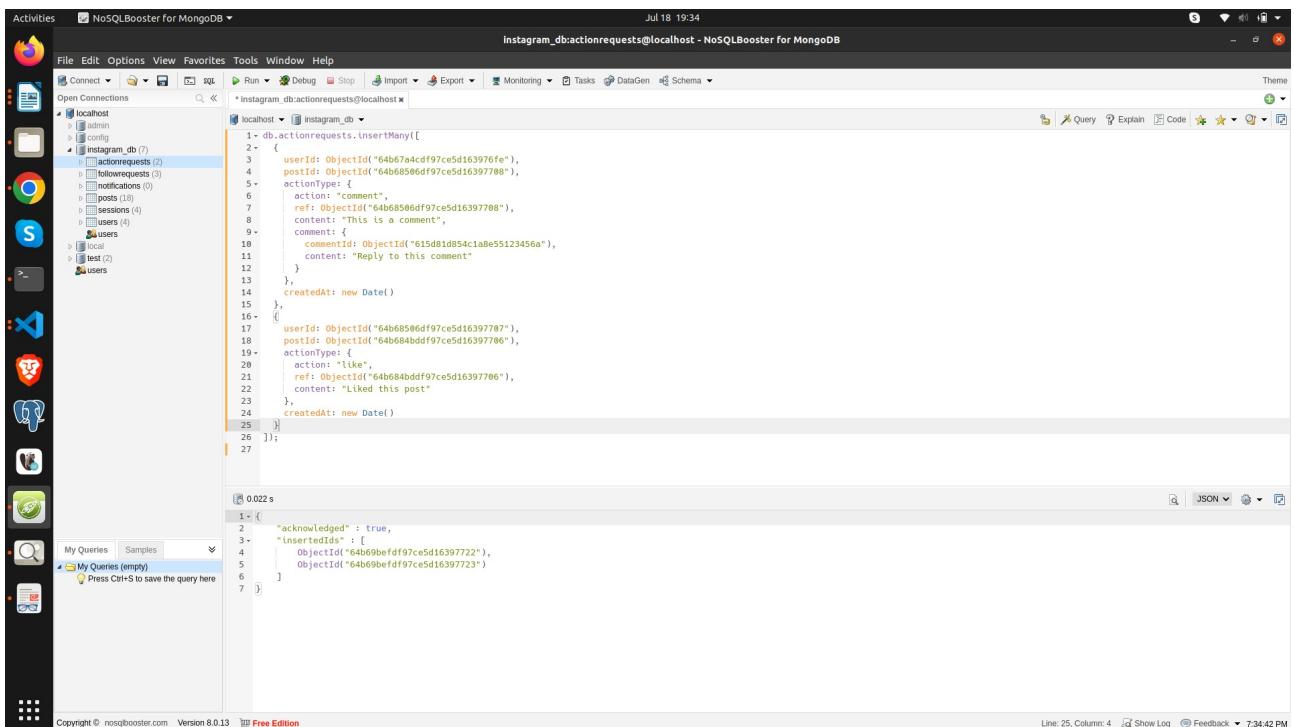
followrequests 0.020 s 3 Docs

_id	senderId	receiverId	createdAt	updatedAt	status	privacy
1 64b69932d97ce5d16397708	64b68506d97ce5d16397708	64b67a4cd97ce5d16397708	18/07/2023, 19:22:50 - 2 minutes ago	18/07/2023, 19:22:50 - 2 minutes ago	pending	true
2 64b69932d97ce5d16397708	64b68506d97ce5d16397708	64b684bdd97ce5d16397708	18/07/2023, 19:22:50 - 2 minutes ago	18/07/2023, 19:22:50 - 2 minutes ago	accepted	false
3 64b69932d97ce5d16397708	64b68506d97ce5d16397708	64b68506d97ce5d16397708	18/07/2023, 19:22:50 - 2 minutes ago	18/07/2023, 19:22:50 - 2 minutes ago	rejected	false

Copyright © nosqbooster.com Version 8.0.13  Free Edition

Line: 1, Column: 24 Show Log Feedback 7:24:57 PM

for inserting in actionRequest schema model:



Activities NoSQLBooster for MongoDB Jul 18 19:34

File Edit Options View Favorites Tools Window Help

localhost Instagram_db (7) Instagram_db (7) sessions (4) users (4) posts (18) notifications (0) followrequests (2) actionrequests (2) admin config

localhost Instagram_db 1 db.actionrequests.insertMany([

```
1- db.actionrequests.insertMany([
2-   {
3-     userId: ObjectId("64b67a4cd97ce5d163976fe"),
4-     postId: ObjectId("64b68506d97ce5d16397788"),
5-     actionType: "comment",
6-     actionId: ObjectId("64b68506d97ce5d16397708"),
7-     ref: ObjectId("64b68506d97ce5d16397708"),
8-     content: "This is a comment",
9-     comment: {
10-       commentId: ObjectId("615d1d8541a0e55123456a"),
11-       content: "Reply to this comment"
12-     }
13-   },
14-   {
15-     createdAt: new Date()
16-   },
17-   {
18-     userId: ObjectId("64b68506d97ce5d16397707"),
19-     postId: ObjectId("64b684bdd97ce5d16397786"),
20-     actionType: "like",
21-     ref: ObjectId("64b684bdd97ce5d16397706"),
22-     content: "Liked this post"
23-   },
24-   {
25-     createdAt: new Date()
26-   }
27- ]);
```

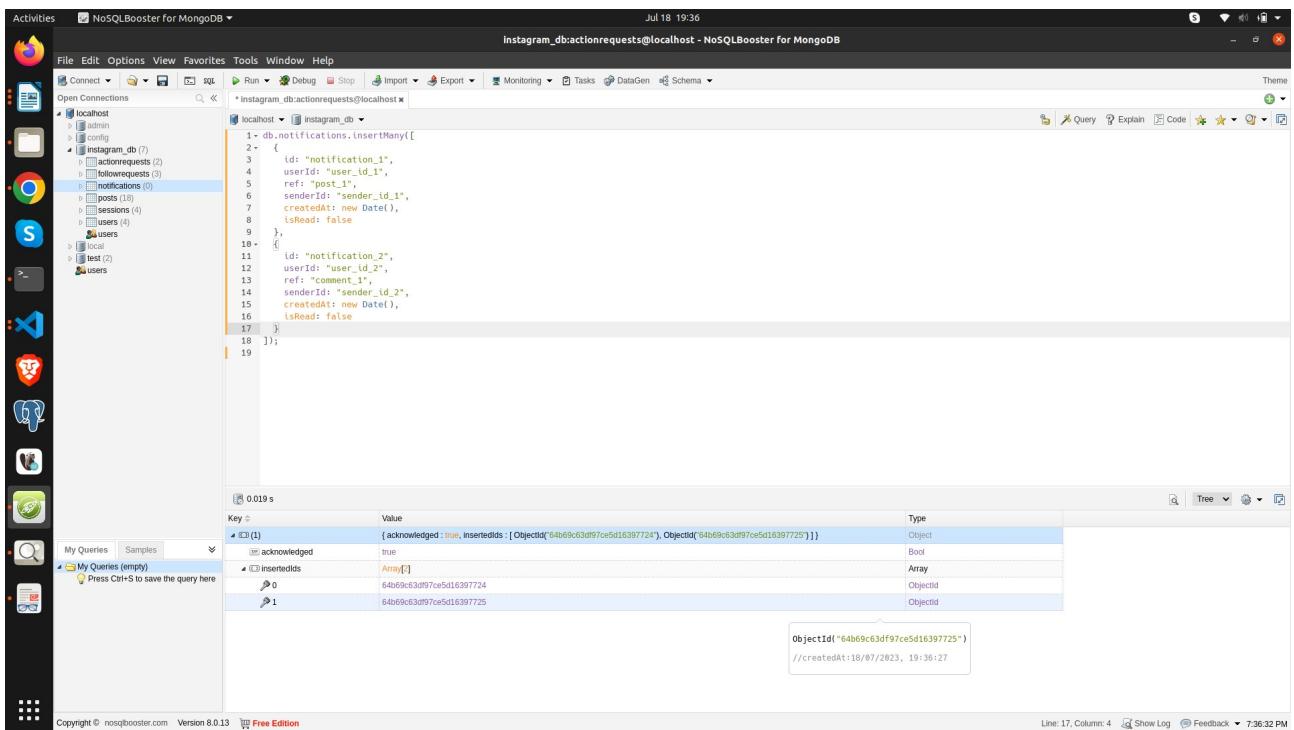
0.022 s

```
1- [
2-   {
3-     "acknowledged": true,
4-     "insertedIds": [
5-       ObjectId("64b69befdf97ce5d16397722"),
6-       ObjectId("64b69befdf97ce5d16397723")
7-     ]
8-   }
9- ]
```

Copyright © nosqbooster.com Version 8.0.13  Free Edition

Line: 25, Column: 4 Show Log Feedback 7:34:42 PM

for inserting in notification schema model:



The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar shows the database structure with the 'notifications' collection selected. The main pane displays a MongoDB query for inserting multiple documents into the 'notifications' collection. The query is as follows:

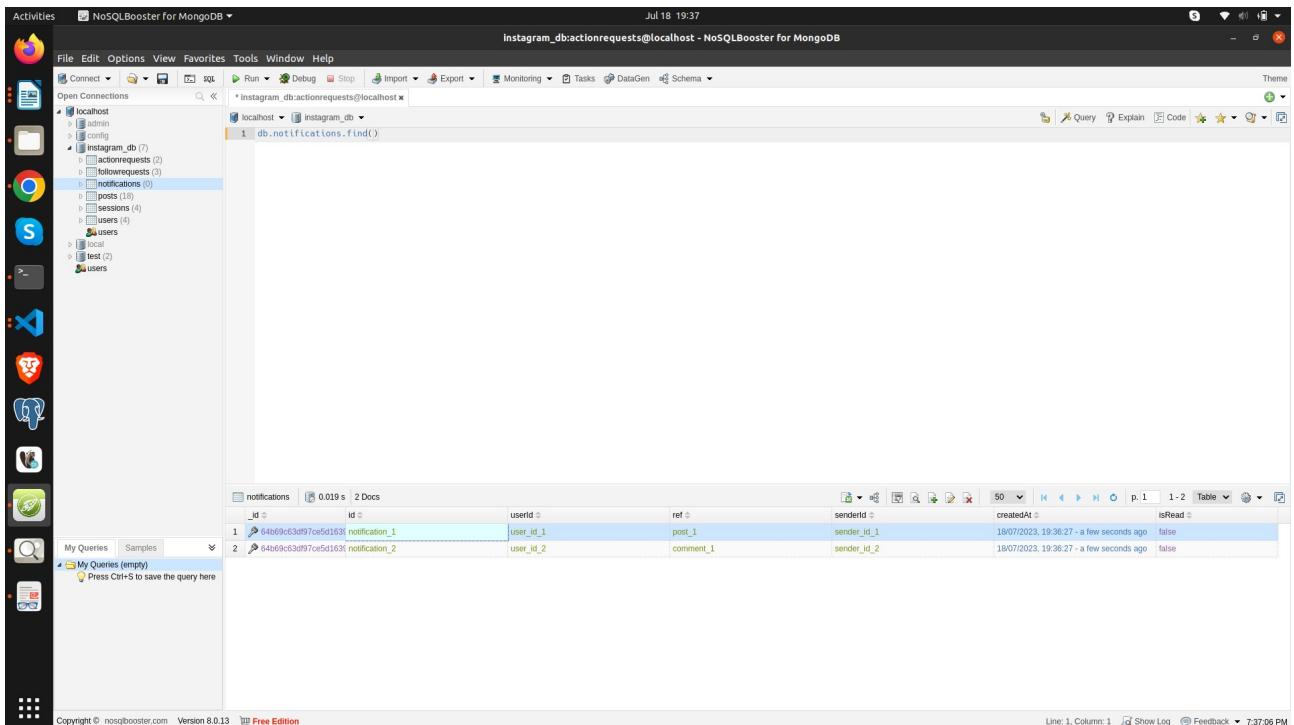
```
1+ db.notifications.insertMany([
2+   {
3+     id: "notification_1",
4+     userId: "user_id_1",
5+     ref: "post_1",
6+     senderId: "sender_id_1",
7+     createdAt: new Date(),
8+     isRead: false
9+   },
10+  {
11+    id: "notification_2",
12+    userId: "user_id_2",
13+    ref: "comment_1",
14+    senderId: "sender_id_2",
15+    createdAt: new Date(),
16+    isRead: false
17+  }
18]);
19
```

The results pane shows the inserted documents with their IDs and creation dates. The first document is:

Key	Value	Type
0 (1)	{ acknowledged: true, insertedIds: [ObjectId("64b69c63df97ce5d16397724"), ObjectId("64b69c63df97ce5d16397725")] }	Object
insertedIds	Array	
0	64b69c63df97ce5d16397724	Objectid
1	64b69c63df97ce5d16397725	Objectid

objectId("64b69c63df97ce5d16397725")
//createdAt:18/07/2023, 19:36:27

for find Query in notifications model:



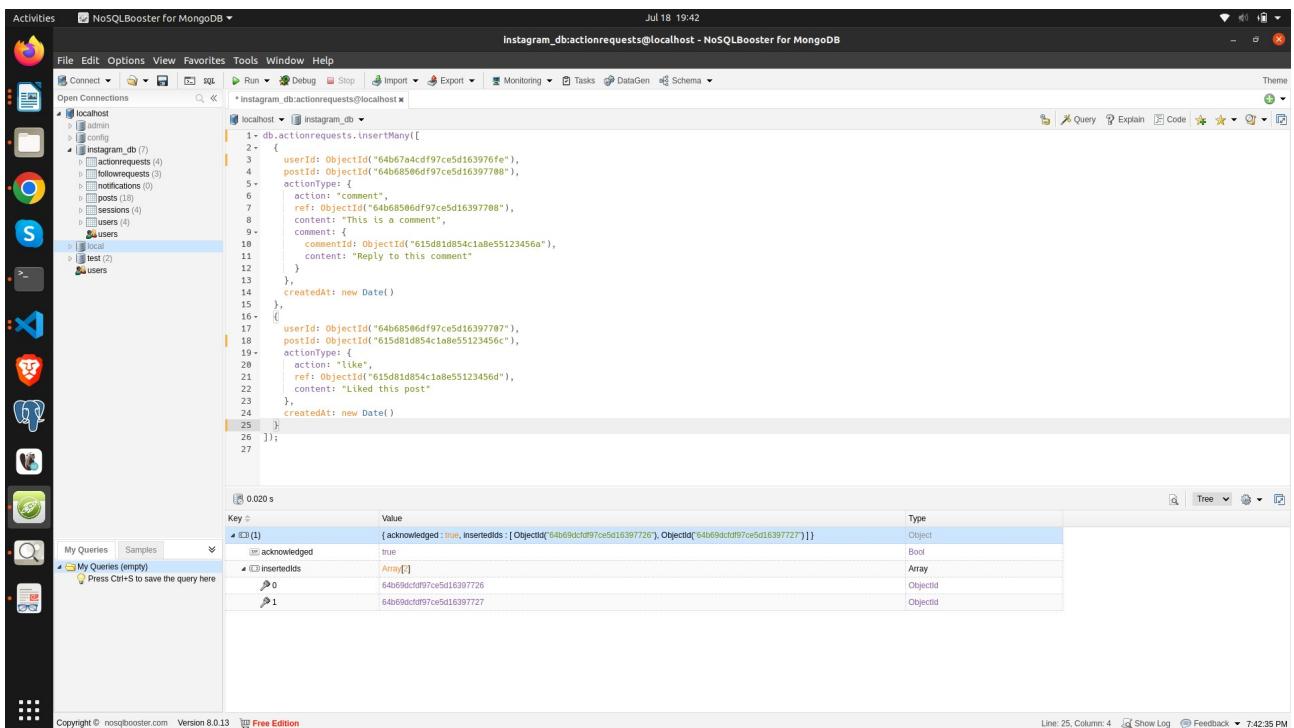
The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar shows the database structure with the 'notifications' collection selected. The main pane displays a MongoDB query for finding documents in the 'notifications' collection. The query is:

```
1 db.notifications.find()
```

The results pane shows the found documents in a table format. There are 2 documents:

notifications	0.019 s 2 Docs																							
<table border="1"><thead><tr><th>_id</th><th>id</th><th>userId</th><th>ref</th><th>senderId</th><th>createdAt</th><th>isRead</th></tr></thead><tbody><tr><td>1</td><td>64b69c63df97ce5d16397724</td><td>notification_1</td><td>user_id_1</td><td>post_1</td><td>sender_id_1</td><td>18/07/2023, 19:36:27 - a few seconds ago</td><td>false</td></tr><tr><td>2</td><td>64b69c63df97ce5d16397725</td><td>notification_2</td><td>user_id_2</td><td>comment_1</td><td>sender_id_2</td><td>18/07/2023, 19:36:27 - a few seconds ago</td><td>false</td></tr></tbody></table>	_id	id	userId	ref	senderId	createdAt	isRead	1	64b69c63df97ce5d16397724	notification_1	user_id_1	post_1	sender_id_1	18/07/2023, 19:36:27 - a few seconds ago	false	2	64b69c63df97ce5d16397725	notification_2	user_id_2	comment_1	sender_id_2	18/07/2023, 19:36:27 - a few seconds ago	false	0.019 s 2 Docs
_id	id	userId	ref	senderId	createdAt	isRead																		
1	64b69c63df97ce5d16397724	notification_1	user_id_1	post_1	sender_id_1	18/07/2023, 19:36:27 - a few seconds ago	false																	
2	64b69c63df97ce5d16397725	notification_2	user_id_2	comment_1	sender_id_2	18/07/2023, 19:36:27 - a few seconds ago	false																	

for inserting in ActionResponse schema model:

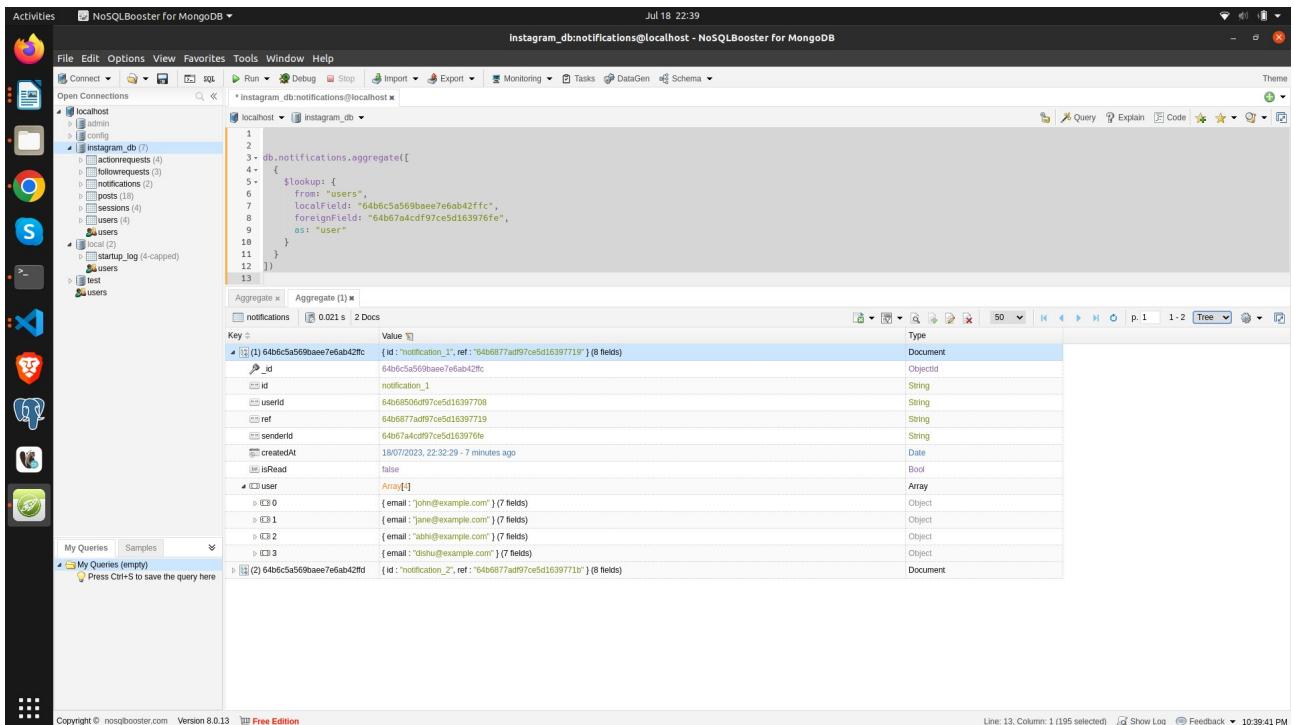


```
1 db.actionrequests.insertMany([
2 {
3   userId: ObjectId('64b67a4cdf97ce5d163976fe'),
4   postId: ObjectId('64b6856d9f97ce5d16397788'),
5   actionType: 'comment',
6   ref: ObjectId('64b68596df97ce5d16397708'),
7   content: "This is a comment",
8   comment: {
9     commentId: ObjectId('615d81d8541a8e55123456a'),
10    content: "Reply to this comment"
11   }
12 },
13 {
14   createdAt: new Date()
15 },
16 {
17   userId: ObjectId('64b6856d9f97ce5d16397707'),
18   postId: ObjectId('615d81d8541a8e55123456c'),
19   actionType: 'like',
20   ref: ObjectId('615d81d8541a8e55123456d'),
21   content: "Liked this post"
22 },
23 {
24   createdAt: new Date()
25 }
26 ])
27
```

0.020 s

Key	Value	Type
0 (1)	{ acknowledged: true, insertedIds: [ObjectId('64b69dd97ce5d16397726'), ObjectId('64b69dd97ce5d16397727')] }	Object
insertedIds	Array	
0	64b69dd97ce5d16397726	Objectid
1	64b69dd97ce5d16397727	Objectid

for pipeline lookup

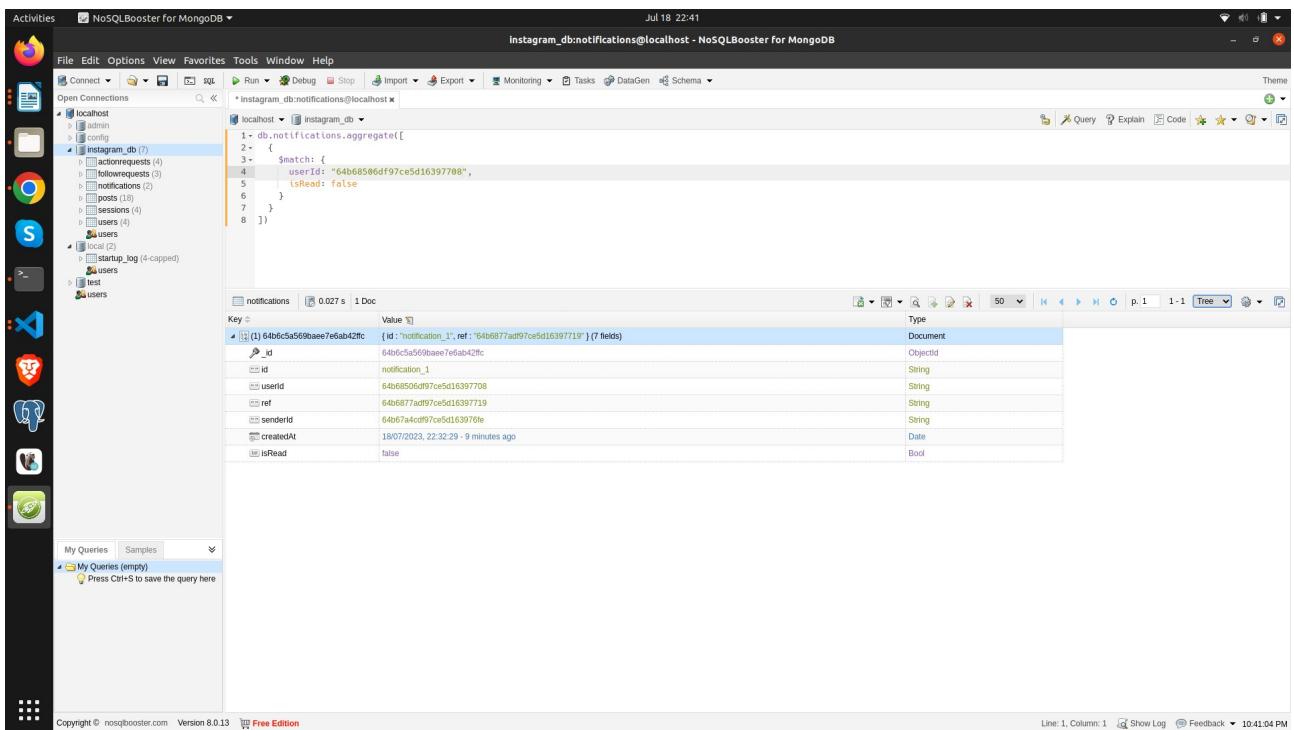


```
1 db.notifications.aggregate([
2 {
3   $lookup: {
4     from: "users",
5     localField: "64b6c5a569baee7e6ab42ffc",
6     foreignField: "64b67a4cdf97ce5d163976fe",
7     as: "user"
8   }
9 }
10 ])
11
12 ])
```

0.021 s | 2 Docs

Key	Value	Type
0 (1) 64b6c5a569baee7e6ab42ffc	{ id: "notification_1", ref: "64b6877ad97ce5d16397719" } (8 fields)	Document
__id	64b6c5a569baee7e6ab42ffc	Objectid
__id	notification_1	String
userId	64b6856d9f97ce5d16397708	String
ref	64b6877ad97ce5d16397719	String
senderId	64b67a4cdf97ce5d163976fe	String
createdAt	18/07/2023, 22:32:29 - 7 minutes ago	Date
isRead	false	Bool
__use	Array	
0	{ email: "john@example.com" } (7 fields)	Object
1	{ email: "jane@example.com" } (7 fields)	Object
2	{ email: "athir@example.com" } (7 fields)	Object
3	{ email: "dithu@example.com" } (7 fields)	Object
0 (2) 64b6c5a569baee7e6ab42fd	{ id: "notification_2", ref: "64b6877ad97ce5d16397718" } (8 fields)	Document

for \$match



Activities NoSQLBooster for MongoDB ▾ Jul 18 22:41

File Edit Options View Favorites Tools Window Help

Open Connections

- localhost
- Instagram_db (7)
- actionrequests (4)
- followerrequests (3)
- notifications (2)
- posts (18)
- sessions (4)
- users (4)
- local (2)
- startup_log (4-capped)
- test
- users

localhost ▾ Instagram_db ▾

```
1 - db.notifications.aggregate([
2 - {
3 -   $match: {
4 -     userId: "64b68586df97ce5d16397708",
5 -     isRead: false
6 -   }
7 - }
8 - ])
```

notifications 0.027 s 1 Doc

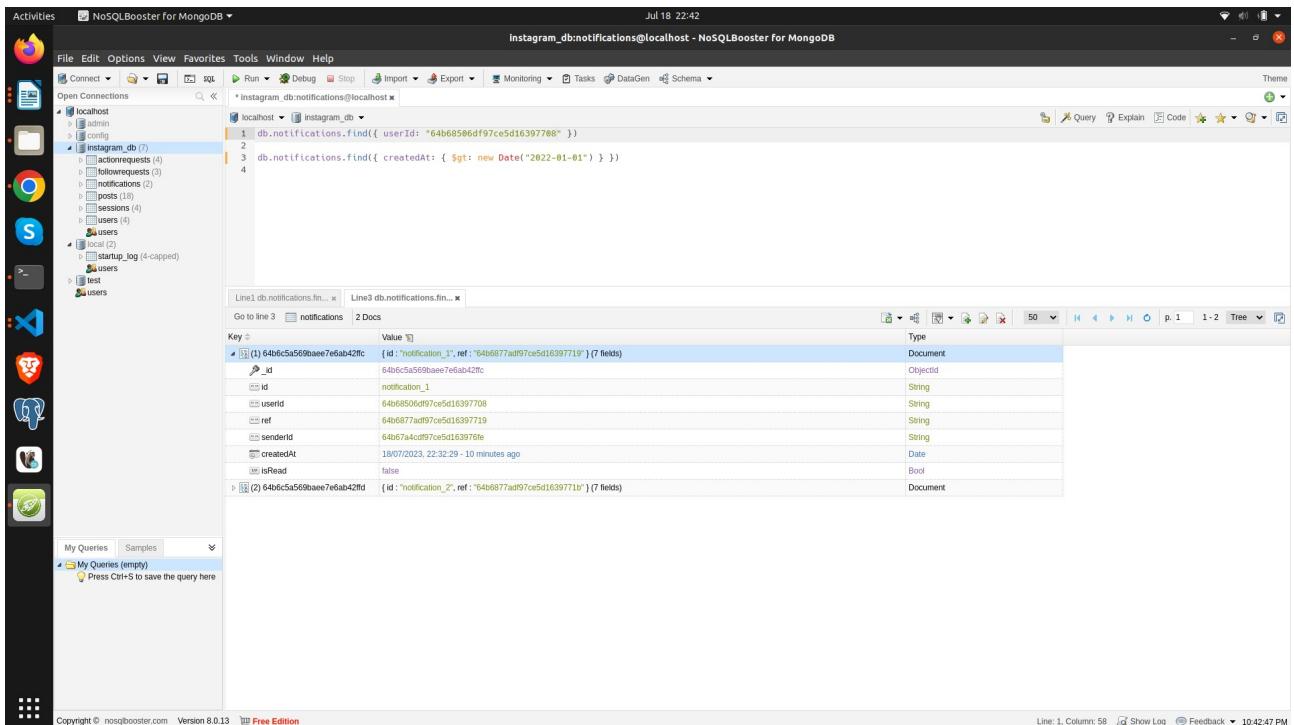
Key	Type
1 (1) 64b6c5a569baee7e6ab42fc { id: "notification_1", ref: "64b6877ad97ce5d16397719" } (7 fields)	Document
__id	ObjectId
id	String
notification_1	String
userId	String
ref	String
senderId	String
createdAt	Date
isRead	Bool

My Queries Samples ▾ My Queries (empty) Press Ctrl+S to save the query here

Copyright © nosqbooster.com Version 8.0.13  Free Edition

Line: 1, Column: 1 Show Log Feedback 10:41:04 PM

for \$gt



Activities NoSQLBooster for MongoDB ▾ Jul 18 22:42

File Edit Options View Favorites Tools Window Help

Open Connections

- localhost
- Instagram_db (7)
- actionrequests (4)
- followerrequests (3)
- notifications (2)
- posts (18)
- sessions (4)
- users (4)
- local (2)
- startup_log (4-capped)
- test
- users

localhost ▾ Instagram_db ▾

```
1 db.notifications.find({ userId: "64b68586df97ce5d16397708" })
2
3 db.notifications.find({ createdAt: { $gt: new Date("2022-01-01") } })
4
```

Line1 db.notifications.fin... Line3 db.notifications.fin... ▾

Go to line 3 notifications 2 Docs

Key	Type
1 (1) 64b6c5a569baee7e6ab42fc { id: "notification_1", ref: "64b6877ad97ce5d16397719" } (7 fields)	Document
__id	ObjectId
id	String
notification_1	String
userId	String
ref	String
senderId	String
createdAt	Date
isRead	Bool

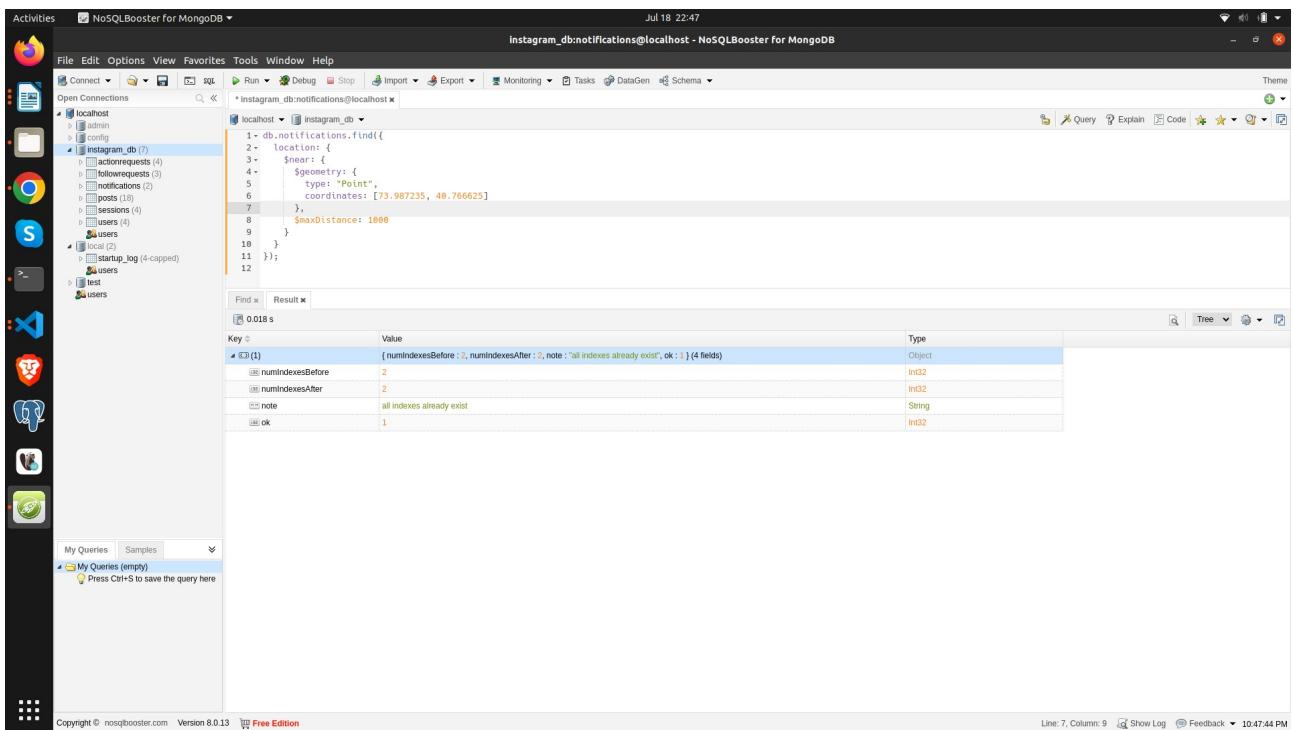
2 (2) 64b6c5a569baee7e6ab42fd { id: "notification_2", ref: "64b6877ad97ce5d16397719" } (7 fields)

My Queries Samples ▾ My Queries (empty) Press Ctrl+S to save the query here

Copyright © nosqbooster.com Version 8.0.13  Free Edition

Line: 1, Column: 58 Show Log Feedback 10:42:47 PM

for geometry/geonear



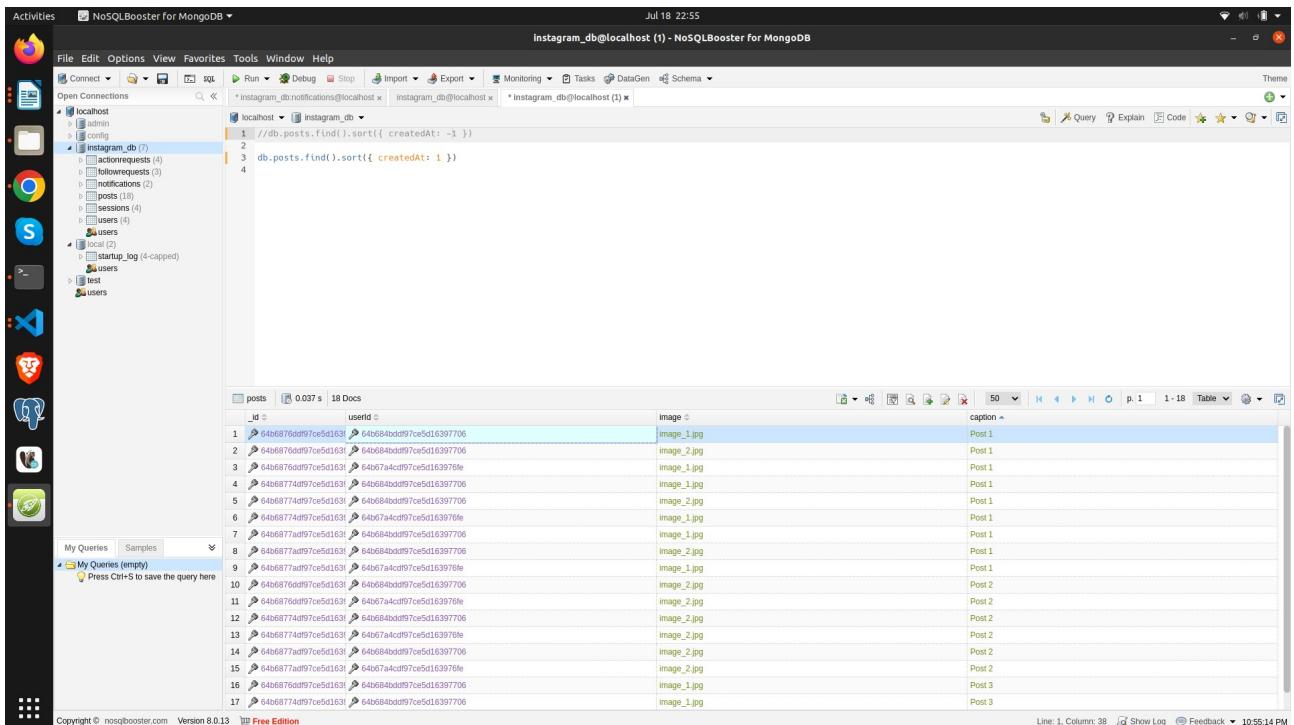
The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar lists databases: admin, config, Instagram_db (selected), and local (2). The Instagram_db section shows collections: actionrequests (4), followerrequests (3), notifications (2), posts (18), sessions (4), and users (4). The posts collection is expanded, showing documents with fields like _id, user_id, and image. A query editor at the top has the following code:

```
localhost: > db.notifications.find({
  2-   location: {
  3-     $near: {
  4-       $geometry: {
  5-         type: "Point",
  6-         coordinates: [73.987235, 40.766625]
  7-       }
  8-       $maxDistance: 1000
  9-     }
 10-   }
 11- });
 12- );
```

The results table shows one document with the following fields:

Key	Value	Type
0 (1)	{ numIndexesBefore: 2, numIndexesAfter: 2, note: "all indexes already exist", ok: 1 } (4 fields)	Object
1 numIndexesBefore	2	Int32
2 numIndexesAfter	2	Int32
3 note	all indexes already exist	String
4 ok	1	Int32

for sorting



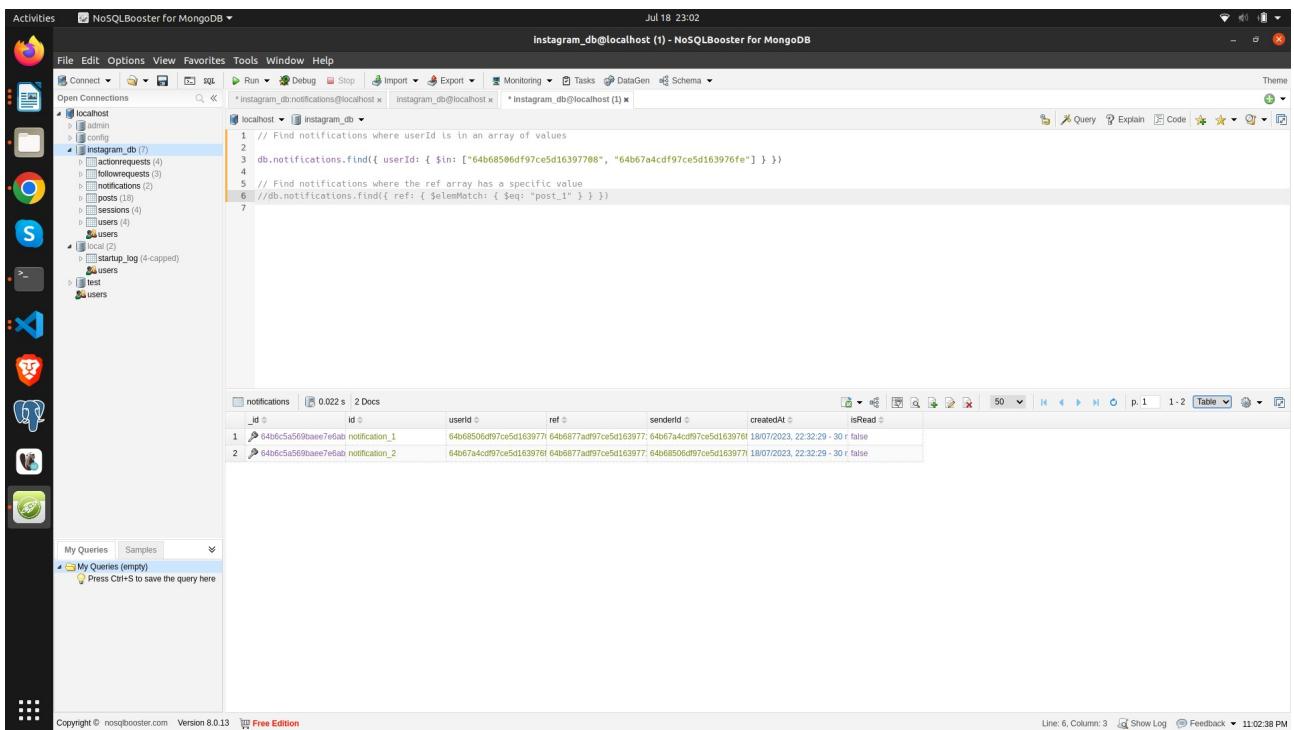
The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar lists databases: admin, config, Instagram_db (selected), and local (2). The Instagram_db section shows collections: actionrequests (4), followerrequests (3), notifications (2), posts (18), sessions (4), and users (4). The posts collection is expanded, showing documents with fields like _id, user_id, and image. A query editor at the top has the following code:

```
localhost: > //db.posts.find().sort({ createdAt: -1 })
 2- 
 3- db.posts.find().sort({ createdAt: 1 })
 4- 
```

The results table shows 18 documents from the posts collection, sorted by createdAt in descending order (Post 1 at the top, Post 3 at the bottom). The columns are _id, user_id, image, and caption.

_id	user_id	image	caption
1	6468766d97ce5d16397706	image_1.jpg	Post 1
2	6468766d97ce5d16397706	image_2.jpg	Post 1
3	6468766d97ce5d163976fe	image_1.jpg	Post 1
4	6468774d97ce5d16397706	image_1.jpg	Post 1
5	6468774d97ce5d16397706	image_2.jpg	Post 1
6	6468774d97ce5d163976fe	image_1.jpg	Post 1
7	646877ad97ce5d16397706	image_1.jpg	Post 1
8	646877ad97ce5d16397706	image_2.jpg	Post 1
9	646877ad97ce5d163976fe	image_1.jpg	Post 1
10	646877ad97ce5d16397706	image_2.jpg	Post 2
11	646877ad97ce5d163976fe	image_1.jpg	Post 2
12	646877ad97ce5d16397706	image_2.jpg	Post 2
13	646877ad97ce5d163976fe	image_1.jpg	Post 2
14	646877ad97ce5d16397706	image_2.jpg	Post 2
15	646877ad97ce5d163976fe	image_1.jpg	Post 2
16	646877ad97ce5d16397706	image_2.jpg	Post 3
17	646877ad97ce5d163976fe	image_1.jpg	Post 3

for array operator (\$in)



The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar lists databases: admin, config, Instagram_db (selected), actionrequests, followerrequests, notifications, posts, users, local, startup_log, and test. The notifications collection is selected in the main pane. A query editor window is open with the following code:

```
1 // Find notifications where userId is in an array of values
2
3 db.notifications.find({ userId: { $in: ["64b68506df97ce5d16397708", "64b67a4cdf97ce5d163976fe"] } })
4
5 // Find notifications where the ref array has a specific value
6 db.notifications.find({ ref: { $elemMatch: { $eq: "post_1" } } })
7
```

Below the code, a table shows the results of the query:

	_id	id	userId	ref	senderId	createdAt	isRead
1	64b6c5ab9b0aee7e6ab	notification_1	64b68506df97ce5d1639771	64b68506df97ce5d1639771	64b67a4cdf97ce5d163976f1	18/07/2023, 22:32:29 - 30	r false
2	64b6c5a569b0aee7e6ab	notification_2	64b68506df97ce5d1639701	64b68506df97ce5d1639701	64b67a4cdf97ce5d1639771	18/07/2023, 22:32:29 - 30	r false

At the bottom, the status bar shows: Copyright © nosqlbooster.com Version 8.0.13  Free Edition. Line: 6, Column: 3. Show Log. Feedback. 11:02:38 PM.

