Table of Contents

MongoDB

1 Document-based data modeling representation

Customer Collection embedded with Address

```
Customer {
  _id:
  First_Name:
  Last_Name:
  Email:
  Customer_Status:
  Username:
  Password:
  Phone:
  Address: {
    address:
    City:
    Postal_Code:
  }
}
```

Inventory Collection

```
Inventory {
_id:
Stock_Left:
Film_ID:
```

}

Staff collection

Staff {

  _id:

  First_Name:

  Last_Name:

  Email:

  Username:

  Password:

}

Date collection

Date {

  _id:

  Holiday_Date:

  Holiday_Name:

  Discount:

}

Film collection embedded with Actor, Category, Language, and Rating

Film {

  _id:

  Title:

  Description:

  Release_Year:

```
  Price:
  Actor: {
    first_name:
    last_name:
  },
  Category:
  Language:
  Rating:
}
```

Payment collection

```
Payment {
_id:
amount:
payment_date:
Late_Fee_Charge:
Customer_ID:
Staff_ID:
Rental_ID:
}
```

Rental collection

```
Rental {
_id:
Rental_Date:
Return_Date:
```

Late_Return:

Holiday_Date:

Customer_ID:

Inventory_ID:

Staff_ID:

}

2 Data dictionary

Collection Name	Object Name	Data Type	Unique
Customer	_id	String	Yes
	First_Name	String	
	Last_Name	String	
	Email	String	Yes
	Phone	String	Yes
	Customer_Status	Boolean	
	Username	String	
	Password	String	
	Address	Object	
Address	address	String	
	City	String	
	Postal_Code	String	
Inventory	_id	String	Yes
	Stock_Left	Integer	
	Film_ID	String (Reference)	
Staff	_id	String	Yes
	First_Name	String	
	Last_Name	String	
	Email	String	Yes
	Username	String	
	Password	String	
Date	_id	String	Yes
	Holiday_Date	Date	
	Holiday_Name	String	Yes

	Discount	Double	
Film	_id	String	Yes
	Title	String	Yes
	Description	String	
	Release_Year	Integer	
	Price	Double	
	Actor	Object	
	Category	String	
	Language	String	
	Rating	String	
Payment	_id	String	Yes
	amount	Double	
	payment_date	Date	
	Late_Fee_Charge	Double	
	Customer_ID	String (Reference)	
	Staff_ID	String (Reference)	
	Rental_ID	String (Reference)	
Rental	_id	String	Yes
	Rental_Date	Date	
	Return_Date	Date	
	Late_Return	Integer	
	Holiday_Date	String (Reference)	
	Customer_ID	String (Reference)	
	Inventory_ID	String (Reference)	
	Staff_ID	String (Reference)	

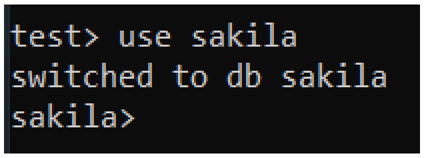
3 Create database

Create the Database

Use the use command to create and switch to the desired database, and the name of the database is sakila.

```
test> use sakila

switched to db sakila
```

sakila>

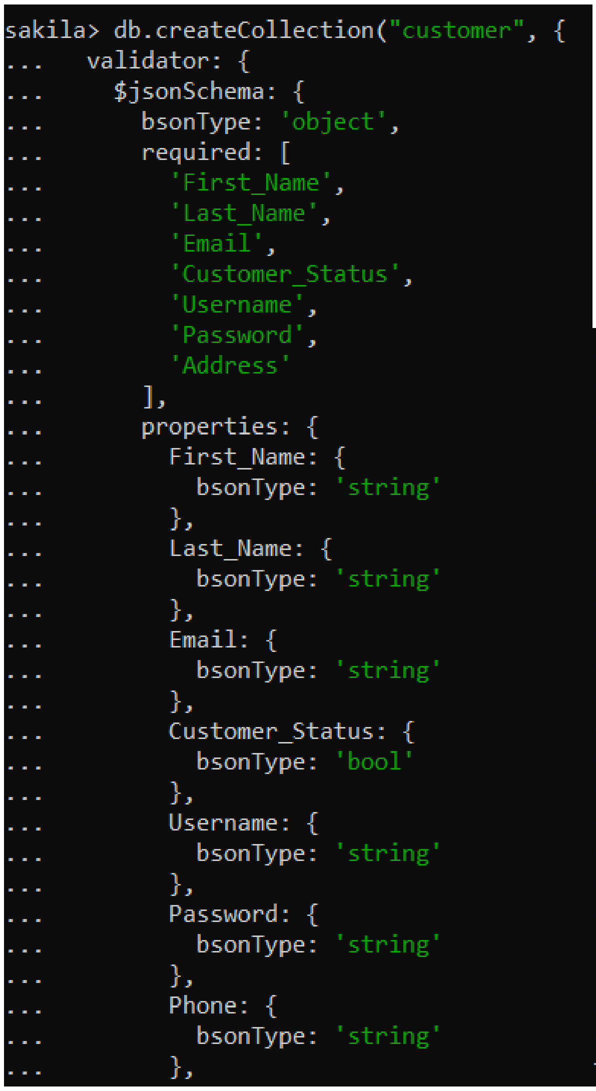

```
test> use sakila
switched to db sakila
sakila>
```

Create Collections

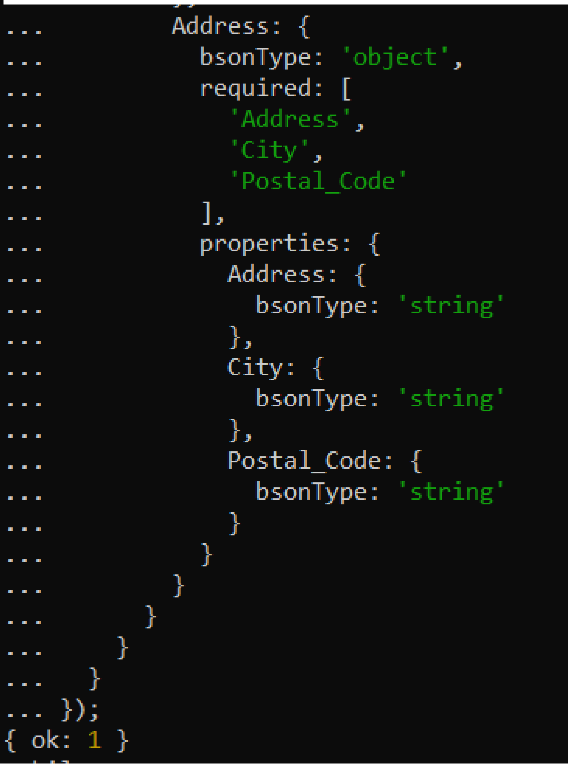
Customer Collection embedded with Address

```
db.createCollection("customer", {
 validator: {
 $jsonSchema: {
  bsonType: 'object',
  required: [
   'First_Name',
   'Last_Name',
   'Email',
   'Customer_Status',
   'Username',
   'Password',
   'Address'
  ],
  properties: {
   First_Name: {
    bsonType: 'string'
   },
   Last_Name: {
    bsonType: 'string'
   },
   Email: {
    bsonType: 'string'
   },
   Customer_Status: {
    bsonType: 'bool'
   },
   Username: {
    bsonType: 'string'
   },
   Password: {
    bsonType: 'string'
   },
   Phone: {
    bsonType: 'string'
   },
   Address: {
    bsonType: 'object',
    required: [
     'Address',
     'City',
     'Postal_Code'
    ],
    properties: {
     Address: {
      bsonType: 'string'
     },
     City: {
      bsonType: 'string'
     },
     Postal_Code: {
      bsonType: 'string'
     }
```

```
    }
   }
  }
 }
}});
```



```
sakila> db.createCollection("customer", {
...   validator: {
...     $jsonSchema: {
...       bsonType: 'object',
...       required: [
...         'First_Name',
...         'Last_Name',
...         'Email',
...         'Customer_Status',
...         'Username',
...         'Password',
...         'Address'
...       ],
...       properties: {
...         First_Name: {
...           bsonType: 'string'
...         },
...         Last_Name: {
...           bsonType: 'string'
...         },
...         Email: {
...           bsonType: 'string'
...         },
...         Customer_Status: {
...           bsonType: 'bool'
...         },
...         Username: {
...           bsonType: 'string'
...         },
...         Password: {
...           bsonType: 'string'
...         },
...         Phone: {
...           bsonType: 'string'
...         },
```



```
...         Address: {
...           bsonType: 'object',
...           required: [
...             'Address',
...             'City',
...             'Postal_Code'
...           ],
...           properties: {
...             Address: {
...               bsonType: 'string'
...             },
...             City: {
...               bsonType: 'string'
...             },
...             Postal_Code: {
...               bsonType: 'string'
...             }
...           }
...         }
...       }
...     }
...   }
... });
{ ok: 1 }
```

Inventory Collection

```
db.createCollection("Inventory", {
 validator: {
```

$jsonSchema: {

"bsonType": "object",

"title": "inventory",

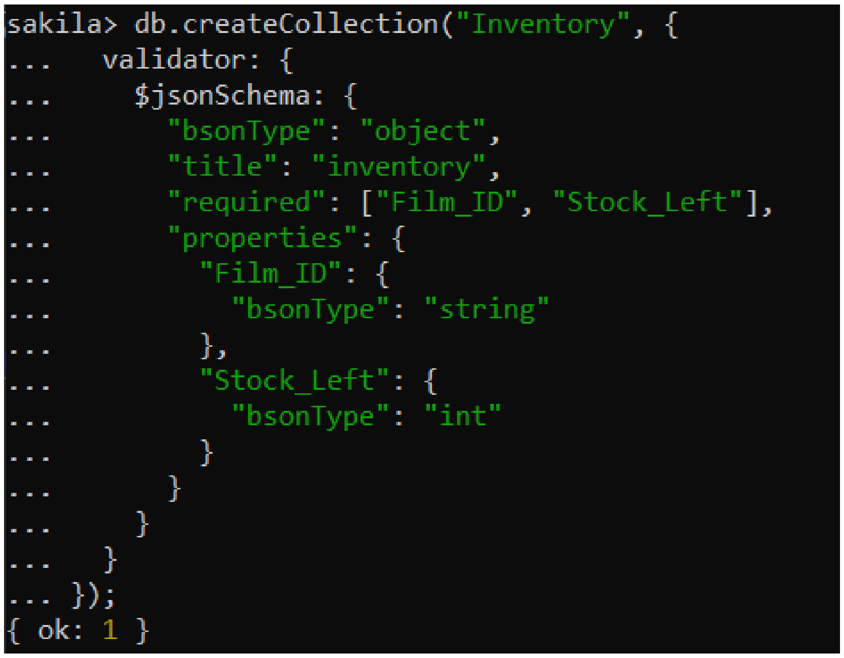
"required": ["Film_ID", "Stock_Left"],

 "properties": {

"Film_ID": { "bsonType": "string" },

"Stock_Left": { "bsonType": "int" }

} } } });



```
sakila> db.createCollection("Inventory", {
...   validator: {
...     $jsonSchema: {
...       "bsonType": "object",
...       "title": "inventory",
...       "required": ["Film_ID", "Stock_Left"],
...       "properties": {
...         "Film_ID": {
...           "bsonType": "string"
...         },
...         "Stock_Left": {
...           "bsonType": "int"
...         }
...       }
...     }
...   }
... });
{ ok: 1 }
```

Staff collection

db.createCollection("staff", {

 validator: { $jsonSchema: {

 bsonType: "object",

 title: "staff",

 required: ["First_Name", "Last_Name", "Email", "Username", "Password"],

 properties: {

First_Name: { bsonType: "string" },

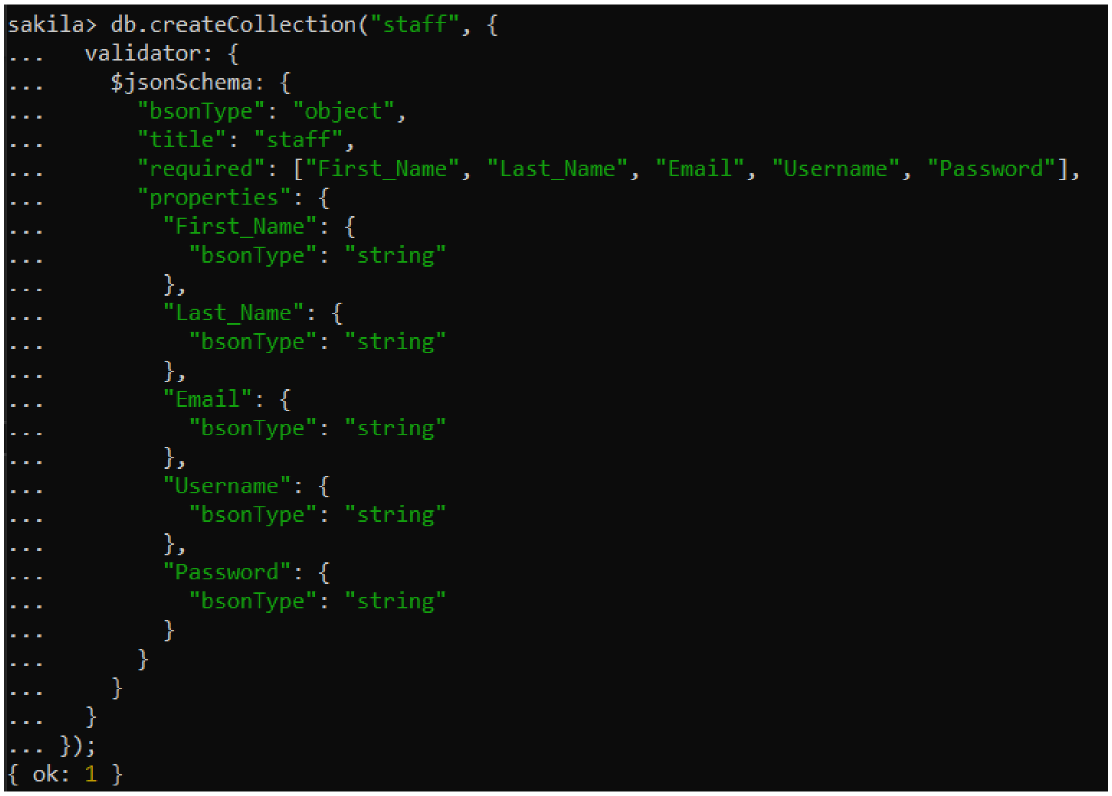
Last_Name: { bsonType: "string" },

Email: { bsonType: "string" },

Username: { bsonType: "string" },

Password: { bsonType: "string" }

} } } });



```
sakila> db.createCollection("staff", {
...   validator: {
...     $jsonSchema: {
...       "bsonType": "object",
...       "title": "staff",
...       "required": ["First_Name", "Last_Name", "Email", "Username", "Password"],
...       "properties": {
...         "First_Name": {
...           "bsonType": "string"
...         },
...         "Last_Name": {
...           "bsonType": "string"
...         },
...         "Email": {
...           "bsonType": "string"
...         },
...         "Username": {
...           "bsonType": "string"
...         },
...         "Password": {
...           "bsonType": "string"
...         }
...       }
...     }
...   }
... });
{ ok: 1 }
```

Date collection

db.createCollection("Date", {

validator: {

$jsonSchema: {

"bsonType": "object",

"title": "Date",

"required": ["Holiday_Date", "Holiday_Name", "Discount"],

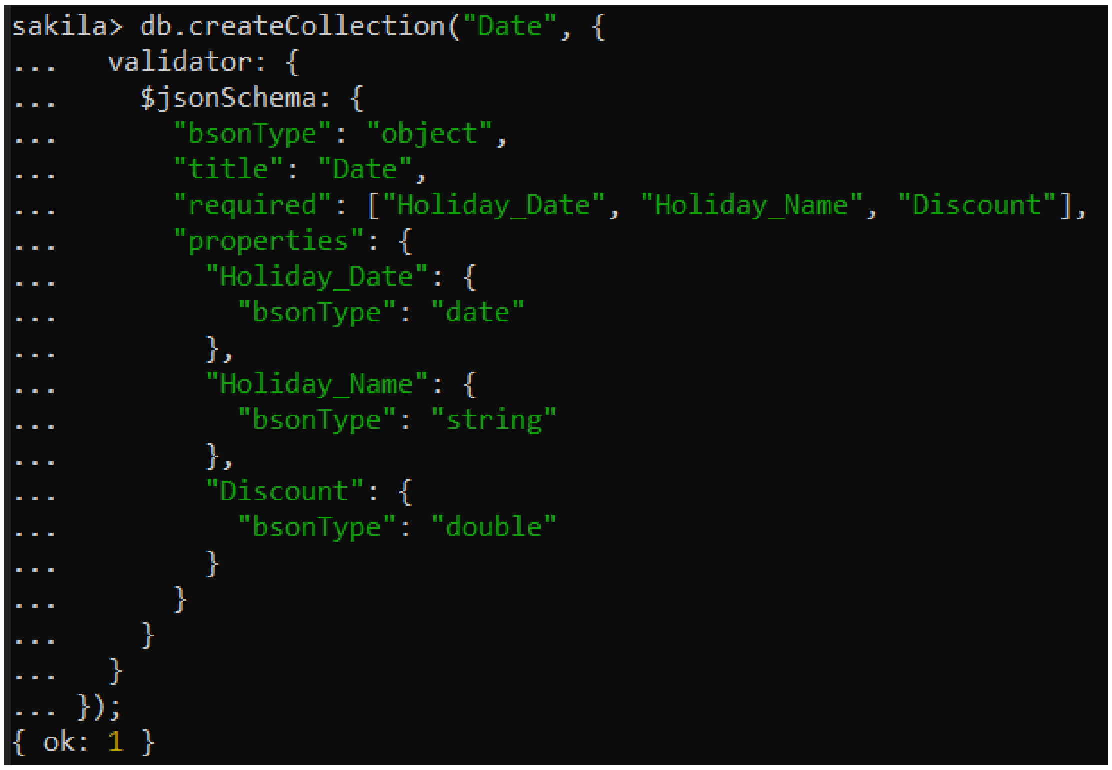
"properties": {

"Holiday_Date": { "bsonType": "date" },

"Holiday_Name": { "bsonType": "string" },

"Discount": { "bsonType": "double" }

} } } });



```
sakila> db.createCollection("Date", {
...   validator: {
...     $jsonSchema: {
...       "bsonType": "object",
...       "title": "Date",
...       "required": ["Holiday_Date", "Holiday_Name", "Discount"],
...       "properties": {
...         "Holiday_Date": {
...           "bsonType": "date"
...         },
...         "Holiday_Name": {
...           "bsonType": "string"
...         },
...         "Discount": {
...           "bsonType": "double"
...         }
...       }
...     }
...   }
... });
{ ok: 1 }
```

Film collection embedded with Actor, Category, Language, and Rating

db.createCollection("Film", {

validator: {

$jsonSchema: { "

bsonType": "object",

"title": "Film",

"required": ["Title", "Description", "Release_Year", "Price", "Actor", "Category", "Language", "Rating"],

"properties": {

"Title": { "bsonType": "string" },

"Description": { "bsonType": "string" },

"Release_Year": { "bsonType": "int" },

 "Price": { "bsonType": "double" },

"Actor": {

 "bsonType": "object",

"required": ["first_name", "last_name"],

"properties": {

"first_name": { "bsonType": "string" },

 "last_name": { "bsonType": "string" }

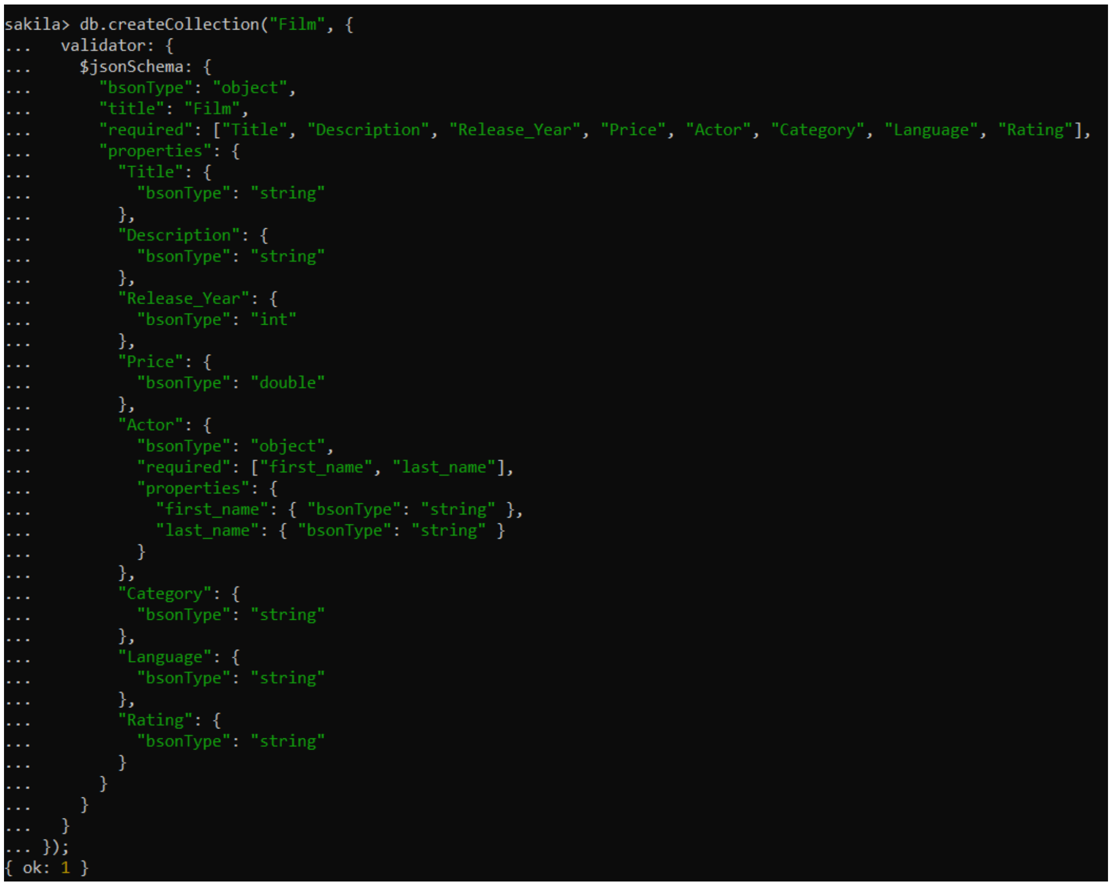
} },

"Category": { "bsonType": "string" },

"Language": { "bsonType": "string" },

"Rating": { "bsonType": "string" }

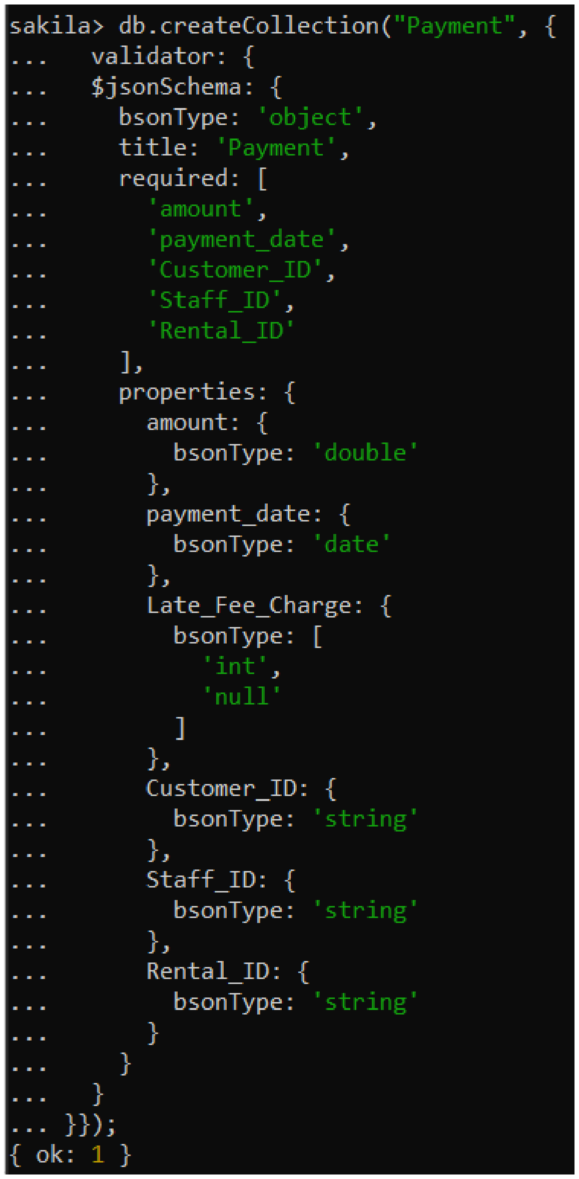
} } } });



```
sakila> db.createCollection("Film", {
...   validator: {
...     $jsonSchema: {
...       "bsonType": "object",
...       "title": "Film",
...       "required": ["Title", "Description", "Release_Year", "Price", "Actor", "Category", "Language", "Rating"],
...       "properties": {
...         "Title": {
...           "bsonType": "string"
...         },
...         "Description": {
...           "bsonType": "string"
...         },
...         "Release_Year": {
...           "bsonType": "int"
...         },
...         "Price": {
...           "bsonType": "double"
...         },
...         "Actor": {
...           "bsonType": "object",
...           "required": ["first_name", "last_name"],
...           "properties": {
...             "first_name": { "bsonType": "string" },
...             "last_name": { "bsonType": "string" }
...           }
...         },
...         "Category": {
...           "bsonType": "string"
...         },
...         "Language": {
...           "bsonType": "string"
...         },
...         "Rating": {
...           "bsonType": "string"
...         }
...       }
...     }
...   }
... });
{ ok: 1 }
```

Payment collection

```
db.createCollection("Payment", {

validator: {

$jsonSchema: {

bsonType: 'object',

title: 'Payment',

required: [ 'amount', 'payment_date', 'Customer_ID', 'Staff_ID', 'Rental_ID' ],

 properties: {

amount: { bsonType: 'double' },

payment_date: { bsonType: 'date' },

 Late_Fee_Charge: { bsonType: [ 'int', 'null' ] },

Customer_ID: { bsonType: 'string' },

Staff_ID: { bsonType: 'string' },

Rental_ID: { bsonType: 'string' }

} } }});
```

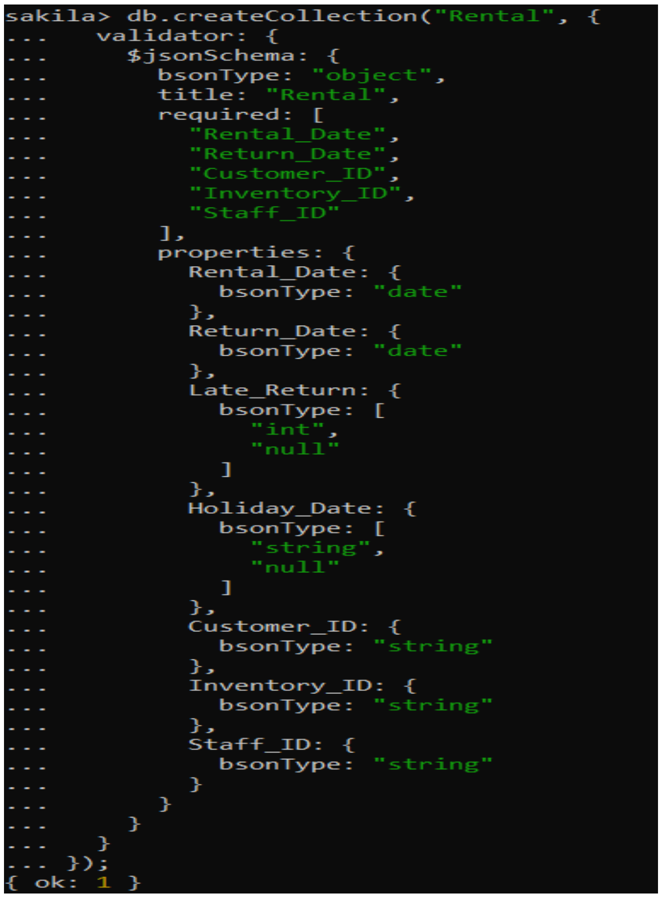


```
sakila> db.createCollection("Payment", {
...   validator: {
...   $jsonSchema: {
...     bsonType: 'object',
...     title: 'Payment',
...     required: [
...       'amount',
...       'payment_date',
...       'Customer_ID',
...       'Staff_ID',
...       'Rental_ID'
...     ],
...     properties: {
...       amount: {
...         bsonType: 'double'
...       },
...       payment_date: {
...         bsonType: 'date'
...       },
...       Late_Fee_Charge: {
...         bsonType: [
...           'int',
...           'null'
...         ]
...       },
...       Customer_ID: {
...         bsonType: 'string'
...       },
...       Staff_ID: {
...         bsonType: 'string'
...       },
...       Rental_ID: {
...         bsonType: 'string'
...       }
...     }
...   }
... }});
{ ok: 1 }
```

Rental collection

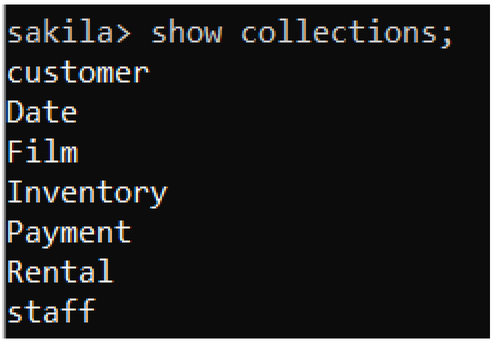
```
db.createCollection("Rental", {

 validator: {

 $jsonSchema: {

  bsonType: 'object',

  title: 'Rental',

  required: [

   'Rental_Date',
   'Return_Date',
   'Customer_ID',
   'Inventory_ID',
   'Staff_ID'
  ],
  properties: {
   Rental_Date: {
    bsonType: 'date'
   },
   Return_Date: {
    bsonType: 'date'
   },
   Late_Return: {
    bsonType: [
     'int',
     'null'
    ]
   },
   Holiday_Date: {
    bsonType: [
     'string',
     'null'
    ]
   },
   Customer_ID: {
    bsonType: 'string'
   },
   Inventory_ID: {
    bsonType: 'string'
```

```
    },
    Staff_ID: {
      bsonType: 'string'
    }
  }
 }
}});
```



```
sakila> db.createCollection("Rental", {
...    validator: {
...      $jsonSchema: {
...        bsonType: "object",
...        title: "Rental",
...        required: [
...          "Rental_Date",
...          "Return_Date",
...          "Customer_ID",
...          "Inventory_ID",
...          "Staff_ID"
...        ],
...        properties: {
...          Rental_Date: {
...            bsonType: "date"
...          },
...          Return_Date: {
...            bsonType: "date"
...          },
...          Late_Return: {
...            bsonType: [
...              "int",
...              "null"
...            ]
...          },
...          Holiday_Date: {
...            bsonType: [
...              "string",
...              "null"
...            ]
...          },
...          Customer_ID: {
...            bsonType: "string"
...          },
...          Inventory_ID: {
...            bsonType: "string"
...          },
...          Staff_ID: {
...            bsonType: "string"
...          }
...        }
...      }
...    }
... });
{ ok: 1 }
```

Verify the Collections

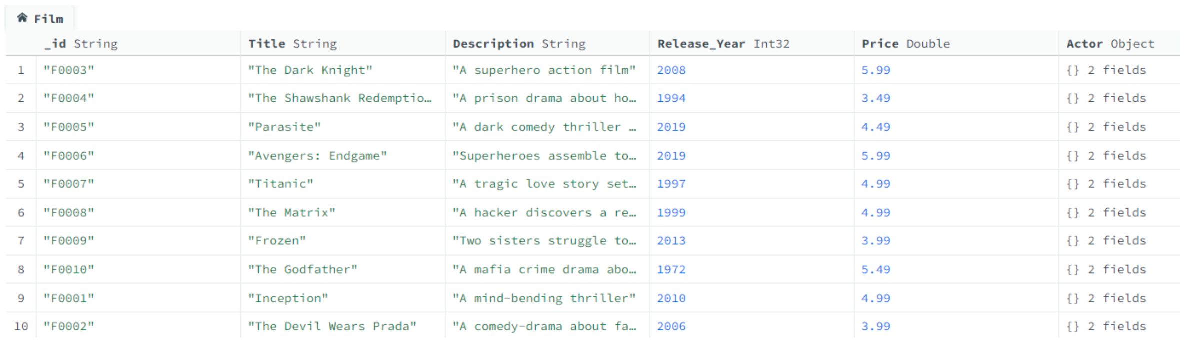


```
sakila> show collections;
customer
Date
Film
Inventory
Payment
Rental
staff
```

4 Enter sample data

Film collection

```
db.Film.insertMany([
 {
  _id: "F0001",
  Title: "Inception",
  Description: "A mind-bending thriller",
  Release_Year: 2010,
  Price: 4.99,
  Actor: {
   first_name: "Leonardo",
   last_name: "DiCaprio"
  },
  Rating: "G",
  Category: "Drama",
  Language: "English"
 },
 {
  _id: "F0002",
  Title: "The Devil Wears Prada",
  Description: "A comedy-drama about fashion",
  Release_Year: 2006,
  Price: 3.99,
  Actor: {
    first_name: "Meryl",
    last_name: "Streep"
  },
  Rating: "G",
  Category: "Drama",
  Language: "English"
},
{
  _id: "F0003",
  Title: "The Dark Knight",
  Description: "A superhero action film",
  Release_Year: 2008,
  Price: 5.99,
  Actor: {
    first_name: "Christian",
    last_name: "Bale"
  },
  Rating: "P13",
  Category: "Action",
  Language: "English"
},
{
  _id: "F0004",
  Title: "The Shawshank Redemption",
  Description: "A prison drama about hope and redemption",
  Release_Year: 1994,
  Price: 3.49,
  Actor: {
    first_name: "Morgan",
    last_name: "Freeman"
  },
  Rating: "P13",
  Category: "Drama",
  Language: "English"
},
{
  _id: "F0005",
  Title: "Parasite",
  Description: "A dark comedy thriller about class struggle",
  Release_Year: 2019,
  Price: 4.49,
  Actor: {
    first_name: "Lee",
    last_name: "Sun-kyun"
  },
  Rating: "P13",
  Category: "Thriller",
  Language: "Korean"
},
{
  _id: "F0006",
  Title: "Avengers: Endgame",
  Description: "Superheroes assemble to stop a global threat",
  Release_Year: 2019,
  Price: 5.99,
  Actor: {
    first_name: "Robert",
    last_name: "Downey Jr."
  },
  Rating: "P13",
  Category: "Action",
  Language: "English"
},
{
  _id: "F0007",
  Title: "Titanic",
  Description: "A tragic love story set on the doomed ship",
  Release_Year: 1997,
  Price: 4.99,
  Actor: {
    first_name: "Leonardo",
    last_name: "DiCaprio"
  },
  Rating: "P13",
  Category: "Drama",
  Language: "English"
},
{
  _id: "F0008",
  Title: "The Matrix",
  Description: "A hacker discovers a reality-bending secret",
  Release_Year: 1999,
  Price: 4.99,
  Actor: {
    first_name: "Keanu",
    last_name: "Reeves"
  },
  Rating: "R",
  Category: "Action",
  Language: "English"
},
{
  _id: "F0009",
  Title: "Frozen",
  Description: "Two sisters struggle to control magical powers",
  Release_Year: 2013,
  Price: 3.99,
  Actor: {
    first_name: "Kristen",
    last_name: "Bell"
  },
  Rating: "G",
  Category: "Fantasy",
  Language: "English"
},
{
  _id: "F0010",
  Title: "The Godfather",
  Description: "A mafia crime drama about family loyalty",
  Release_Year: 1972,
  Price: 5.49,
  Actor: {
   first_name: "Al",
   last_name: "Pacino"
  },
  Rating: "R",
  Category: "Crime",
  Language: "English"
 }
]);
```



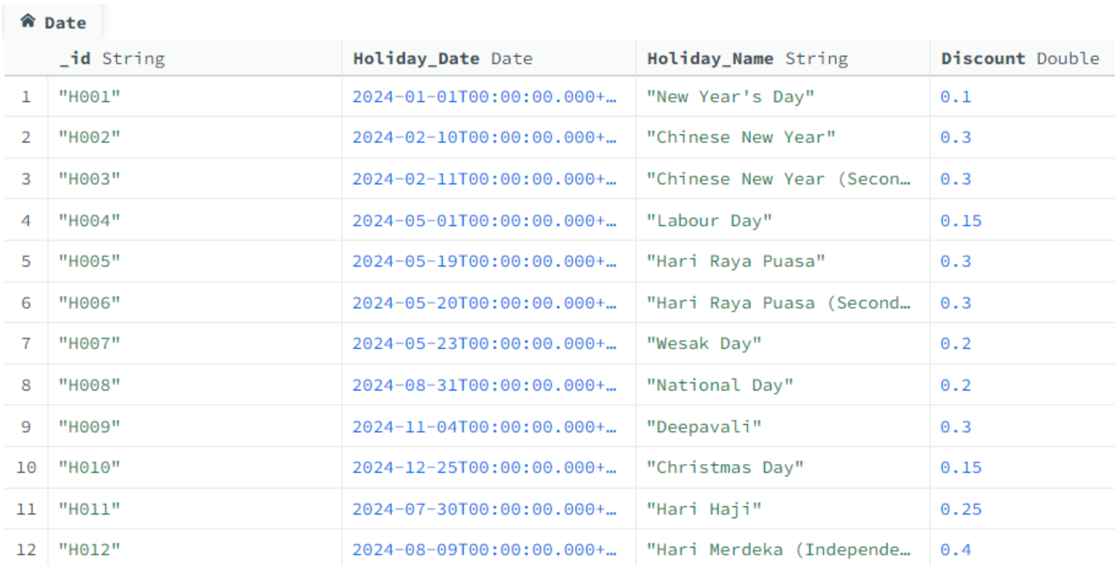
Film

	_id String	Title String	Description String	Release_Year Int32	Price Double	Actor Object
1	"F0003"	"The Dark Knight"	"A superhero action film"	2008	5.99	{} 2 fields
2	"F0004"	"The Shawshank Redemptio…	"A prison drama about ho…	1994	3.49	{} 2 fields
3	"F0005"	"Parasite"	"A dark comedy thriller …	2019	4.49	{} 2 fields
4	"F0006"	"Avengers: Endgame"	"Superheroes assemble to…	2019	5.99	{} 2 fields
5	"F0007"	"Titanic"	"A tragic love story set…	1997	4.99	{} 2 fields
6	"F0008"	"The Matrix"	"A hacker discovers a re…	1999	4.99	{} 2 fields
7	"F0009"	"Frozen"	"Two sisters struggle to…	2013	3.99	{} 2 fields
8	"F0010"	"The Godfather"	"A mafia crime drama abo…	1972	5.49	{} 2 fields
9	"F0001"	"Inception"	"A mind-bending thriller"	2010	4.99	{} 2 fields
10	"F0002"	"The Devil Wears Prada"	"A comedy-drama about fa…	2006	3.99	{} 2 fields

Date collection

```
db.Date.insertMany([
 {
  _id: "H001",
  Holiday_Date: new Date("2024-01-01"),
  Holiday_Name: "New Year's Day",
  Discount: 0.10
 },
 {
  _id: "H002",
  Holiday_Date: new Date("2024-02-10"),
  Holiday_Name: "Chinese New Year",
  Discount: 0.30
```

```
},
{
  _id: "H003",
  Holiday_Date: new Date("2024-02-11"),
  Holiday_Name: "Chinese New Year (Second Day)",
  Discount: 0.30
},
{
  _id: "H004",
  Holiday_Date: new Date("2024-05-01"),
  Holiday_Name: "Labour Day",
  Discount: 0.15
},
{
  _id: "H005",
  Holiday_Date: new Date("2024-05-19"),
  Holiday_Name: "Hari Raya Puasa",
  Discount: 0.30
},
{
  _id: "H006",
  Holiday_Date: new Date("2024-05-20"),
  Holiday_Name: "Hari Raya Puasa (Second Day)",
  Discount: 0.30
},
{
  _id: "H007",
  Holiday_Date: new Date("2024-05-23"),
  Holiday_Name: "Wesak Day",
  Discount: 0.20

},

{

  _id: "H008",

  Holiday_Date: new Date("2024-08-31"),

  Holiday_Name: "National Day",

  Discount: 0.20

},

{

  _id: "H009",

  Holiday_Date: new Date("2024-11-04"),

  Holiday_Name: "Deepavali",

  Discount: 0.30

},

{

  _id: "H010",

  Holiday_Date: new Date("2024-12-25"),

  Holiday_Name: "Christmas Day",

  Discount: 0.15

},

{

  _id: "H011",

  Holiday_Date: new Date("2024-07-30"),

  Holiday_Name: "Hari Haji",

  Discount: 0.25

},

{

  _id: "H012",

  Holiday_Date: new Date("2024-08-09"),
    Holiday_Name: "Hari Merdeka (Independence Day)",
    Discount: 0.40
  }
]);
```

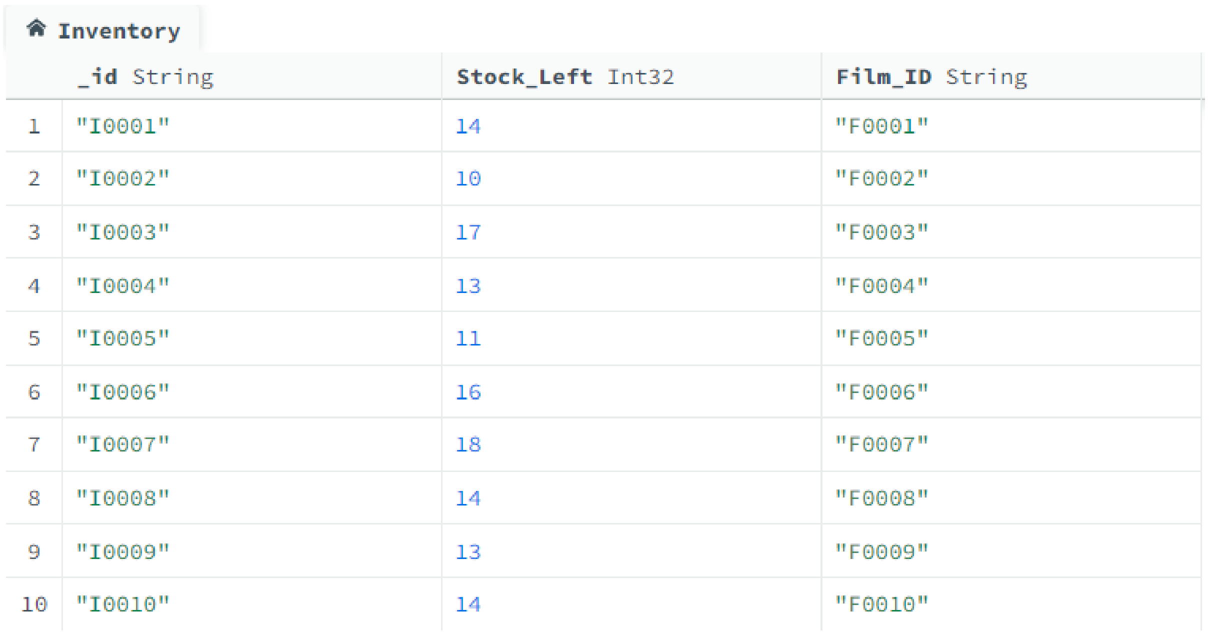


Date

	_id String	Holiday_Date Date	Holiday_Name String	Discount Double
1	"H001"	2024-01-01T00:00:00.000+…	"New Year's Day"	0.1
2	"H002"	2024-02-10T00:00:00.000+…	"Chinese New Year"	0.3
3	"H003"	2024-02-11T00:00:00.000+…	"Chinese New Year (Secon…	0.3
4	"H004"	2024-05-01T00:00:00.000+…	"Labour Day"	0.15
5	"H005"	2024-05-19T00:00:00.000+…	"Hari Raya Puasa"	0.3
6	"H006"	2024-05-20T00:00:00.000+…	"Hari Raya Puasa (Second…	0.3
7	"H007"	2024-05-23T00:00:00.000+…	"Wesak Day"	0.2
8	"H008"	2024-08-31T00:00:00.000+…	"National Day"	0.2
9	"H009"	2024-11-04T00:00:00.000+…	"Deepavali"	0.3
10	"H010"	2024-12-25T00:00:00.000+…	"Christmas Day"	0.15
11	"H011"	2024-07-30T00:00:00.000+…	"Hari Haji"	0.25
12	"H012"	2024-08-09T00:00:00.000+…	"Hari Merdeka (Independe…	0.4

Inventory collection

```
//Function to make sure the film inserted exist in the film table
async function insertInventory(inventory) {
  const filmExists = await db.Film.findOne({ _id: inventory.Film_ID });

  if (filmExists) {
    await db.Inventory.insertOne(inventory);
    console.log("Inventory inserted successfully!");
  } else {
    console.log(`Film_ID ${inventory.Film_ID} does not exist in the Film collection.`);
  }
}

const inventories = [
  { _id: "I0001", Stock_Left: 15, Film_ID: "F0001" },
  { _id: "I0002", Stock_Left: 12, Film_ID: "F0002" },
  { _id: "I0003", Stock_Left: 17, Film_ID: "F0003" },
  { _id: "I0004", Stock_Left: 14, Film_ID: "F0004" },
  { _id: "I0005", Stock_Left: 13, Film_ID: "F0005" },
  { _id: "I0006", Stock_Left: 16, Film_ID: "F0006" },
  { _id: "I0007", Stock_Left: 18, Film_ID: "F0007" },
  { _id: "I0008", Stock_Left: 15, Film_ID: "F0008" },
  { _id: "I0009", Stock_Left: 14, Film_ID: "F0009" },
  { _id: "I0010", Stock_Left: 16, Film_ID: "F0010" }
];

for (const inventory of inventories) {
  insertInventory(inventory);
}
```



Inventory

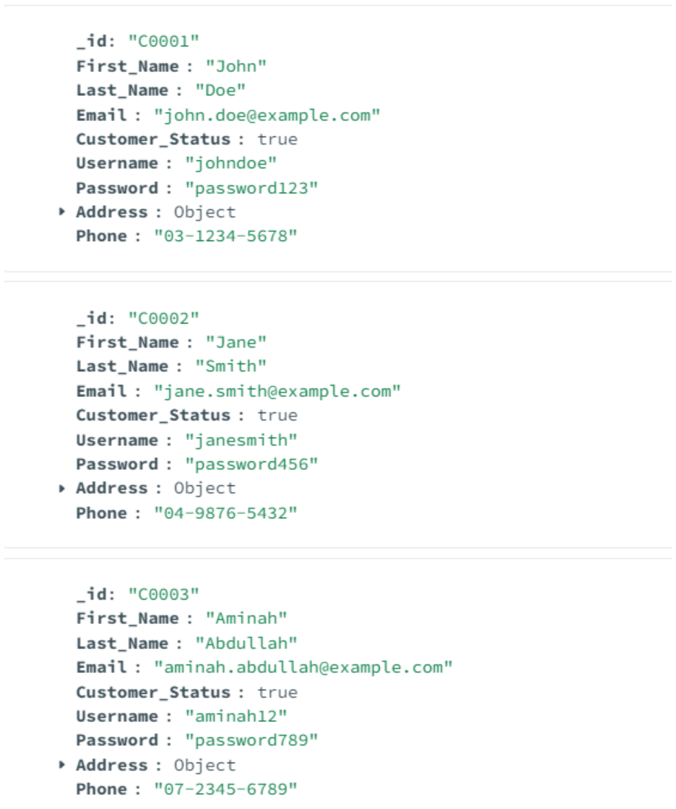
	_id String	Stock_Left Int32	Film_ID String
1	"I0001"	14	"F0001"
2	"I0002"	10	"F0002"
3	"I0003"	17	"F0003"
4	"I0004"	13	"F0004"
5	"I0005"	11	"F0005"
6	"I0006"	16	"F0006"
7	"I0007"	18	"F0007"
8	"I0008"	14	"F0008"
9	"I0009"	13	"F0009"
10	"I0010"	14	"F0010"

Customer collection

```
db.customer.insertMany([
{
 _id: "C0001",
 First_Name: "John",
 Last_Name: "Doe",
 Email: "john.doe@example.com",
 Customer_Status: true,
 Username: "johndoe",
 Password: "password123",
 Phone: "03-1234-5678",
 Address: {
  Address: "123 Elm Street",
  City: "Kuala Lumpur",
  Postal_Code: "50450"
 }
},
{
 _id: "C0002",
 First_Name: "Jane",
 Last_Name: "Smith",
 Email: "jane.smith@example.com",
 Customer_Status: true,
 Username: "janesmith",
 Password: "password456",
 Phone: "04-9876-5432",
 Address: {
  Address: "456 Oak Avenue",
  City: "Penang",
  Postal_Code: "10450"
 }
```

```
},
{
  _id: "C0003",
  First_Name: "Aminah",
  Last_Name: "Abdullah",
  Email: "aminah.abdullah@example.com",
  Customer_Status: true,
  Username: "aminah12",
  Password: "password789",
  Phone: "07-2345-6789",
  Address: {
    Address: "789 Pine Road",
    City: "Johor Bahru",
    Postal_Code: "80000"
  }
},
{
  _id: "C0004",
  First_Name: "Mohd",
  Last_Name: "Ali",
  Email: "mohd.ali@example.com",
  Customer_Status: true,
  Username: "mohdali45",
  Password: "password321",
  Phone: "05-3456-7890",
  Address: {
    Address: "101 Maple Drive",
    City: "Ipoh",
    Postal_Code: "31400"
```

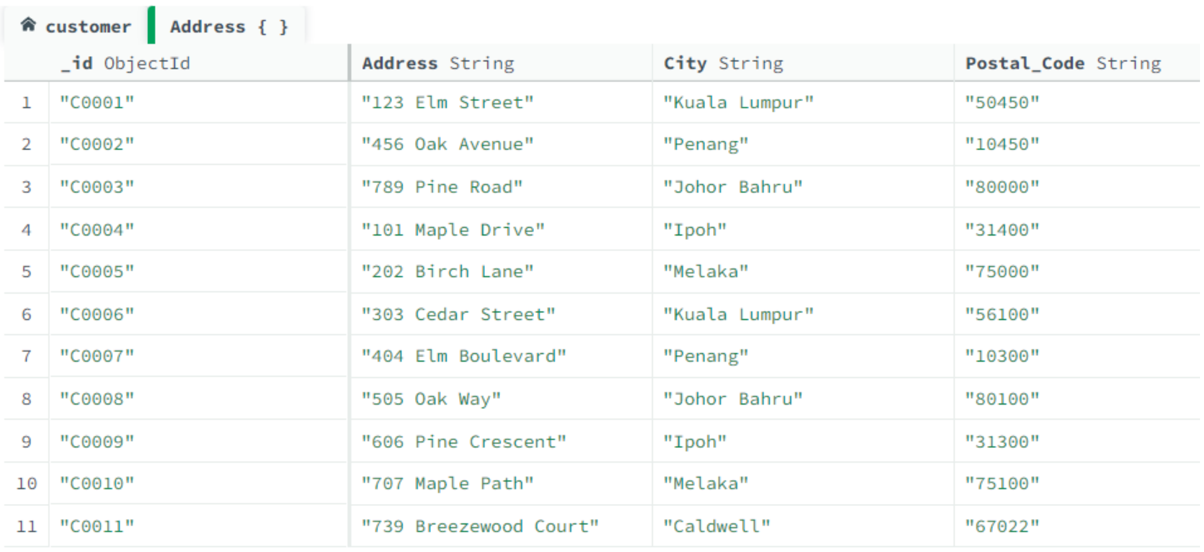
```
 }
},
{
 _id: "C0005",
 First_Name: "Siti",
 Last_Name: "Nur",
 Email: "siti.nur@example.com",
 Customer_Status: true,
 Username: "sitinur",
 Password: "password654",
 Phone: "06-4567-8901",
 Address: {
  Address: "202 Birch Lane",
  City: "Melaka",
  Postal_Code: "75000"
 }
},
{
 _id: "C0006",
 First_Name: "Zahid",
 Last_Name: "Rahman",
 Email: "zahid.rahman@example.com",
 Customer_Status: true,
 Username: "zahidrahman",
 Password: "password234",
 Phone: "03-5678-1234",
 Address: {
  Address: "303 Cedar Street",
  City: "Kuala Lumpur",
  Postal_Code: "56100"
 }
},
{
 _id: "C0007",
 First_Name: "Fatimah",
 Last_Name: "Ismail",
 Email: "fatimah.ismail@example.com",
 Customer_Status: true,
 Username: "fatimah85",
 Password: "password567",
 Phone: "04-6789-1234",
 Address: {
  Address: "404 Elm Boulevard",
  City: "Penang",
  Postal_Code: "10300"
 }
},
{
 _id: "C0008",
 First_Name: "Adam",
 Last_Name: "Tan",
 Email: "adam.tan@example.com",
 Customer_Status: true,
 Username: "adam-tan",
 Password: "password987",
 Phone: "07-7890-1234",
 Address: {
  Address: "505 Oak Way",
  City: "Johor Bahru",
  Postal_Code: "80100"
 }
},
{
 _id: "C0009",
 First_Name: "Isha",
 Last_Name: "Mohamad",
 Email: "isha.mohamad@example.com",
 Customer_Status: true,
 Username: "ishamohamad",
 Password: "password876",
 Phone: "05-8901-2345",
 Address: {
  Address: "606 Pine Crescent",
  City: "Ipoh",
  Postal_Code: "31300"
 }
},
{
 _id: "C0010",
 First_Name: "Ravi",
 Last_Name: "Kumar",
 Email: "ravi.kumar@example.com",
 Customer_Status: true,
 Username: "ravikumar",
 Password: "password543",
 Phone: "06-9012-3456",
 Address: {
   Address: "707 Maple Path",

   City: "Melaka",

   Postal_Code: "75100"

  }

 }

]);
```



```
_id: "C0001"
First_Name : "John"
Last_Name : "Doe"
Email : "john.doe@example.com"
Customer_Status : true
Username : "johndoe"
Password : "password123"
▸ Address : Object
Phone : "03-1234-5678"
```

```
_id: "C0002"
First_Name : "Jane"
Last_Name : "Smith"
Email : "jane.smith@example.com"
Customer_Status : true
Username : "janesmith"
Password : "password456"
▸ Address : Object
Phone : "04-9876-5432"
```

```
_id: "C0003"
First_Name : "Aminah"
Last_Name : "Abdullah"
Email : "aminah.abdullah@example.com"
Customer_Status : true
Username : "aminah12"
Password : "password789"
▸ Address : Object
Phone : "07-2345-6789"
```



customer | Address { }

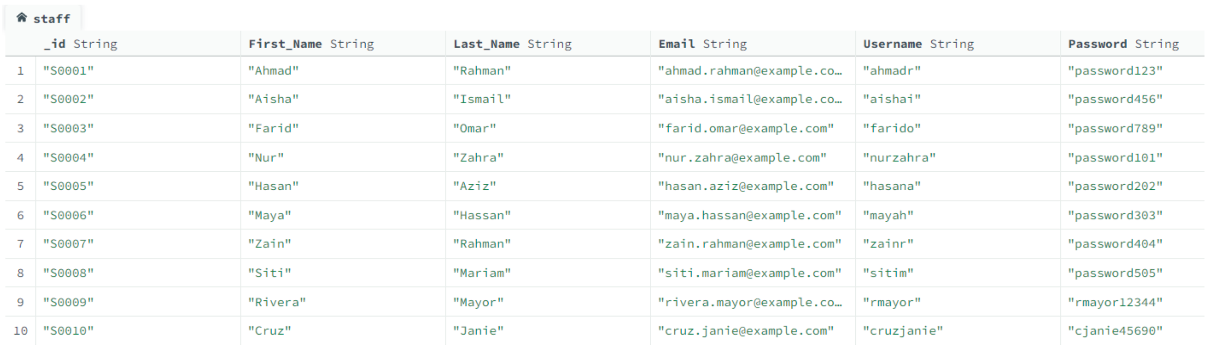
	_id ObjectId	Address String	City String	Postal_Code String
1	"C0001"	"123 Elm Street"	"Kuala Lumpur"	"50450"
2	"C0002"	"456 Oak Avenue"	"Penang"	"10450"
3	"C0003"	"789 Pine Road"	"Johor Bahru"	"80000"
4	"C0004"	"101 Maple Drive"	"Ipoh"	"31400"
5	"C0005"	"202 Birch Lane"	"Melaka"	"75000"
6	"C0006"	"303 Cedar Street"	"Kuala Lumpur"	"56100"
7	"C0007"	"404 Elm Boulevard"	"Penang"	"10300"
8	"C0008"	"505 Oak Way"	"Johor Bahru"	"80100"
9	"C0009"	"606 Pine Crescent"	"Ipoh"	"31300"
10	"C0010"	"707 Maple Path"	"Melaka"	"75100"
11	"C0011"	"739 Breezewood Court"	"Caldwell"	"67022"

Staff collection

```
db.staff.insertMany([

 {

  "_id": "S0001",

  "First_Name": "Ahmad",

  "Last_Name": "Rahman",

  "Email": "ahmad.rahman@example.com",

  "Username": "ahmadr",

  "Password": "password123"

 },

 {

  "_id": "S0002",

  "First_Name": "Aisha",

  "Last_Name": "Ismail",

  "Email": "aisha.ismail@example.com",

  "Username": "aishai",

  "Password": "password456"

 },

 {

  "_id": "S0003",

  "First_Name": "Farid",

  "Last_Name": "Omar",

  "Email": "farid.omar@example.com",

  "Username": "farido",

  "Password": "password789"

 },

 {

  "_id": "S0004",
```

"First_Name": "Nur",
"Last_Name": "Zahra",
"Email": "nur.zahra@example.com",
"Username": "nurzahra",
"Password": "password101"
},
{
"_id": "S0005",
"First_Name": "Hasan",
"Last_Name": "Aziz",
"Email": "hasan.aziz@example.com",
"Username": "hasana",
"Password": "password202"
},
{
"_id": "S0006",
"First_Name": "Maya",
"Last_Name": "Hassan",
"Email": "maya.hassan@example.com",
"Username": "mayah",
"Password": "password303"
},
{
"_id": "S0007",
"First_Name": "Zain",
"Last_Name": "Rahman",
"Email": "zain.rahman@example.com",
"Username": "zainr",
"Password": "password404"

```
 },
 {
  "_id": "S0008",
  "First_Name": "Siti",
  "Last_Name": "Mariam",
  "Email": "siti.mariam@example.com",
  "Username": "sitim",
  "Password": "password505"
 },
 {
  "_id": "S0009",
  "First_Name": "Rivera",
  "Last_Name": "Mayor",
  "Email": "rivera.mayor@example.com",
  "Username": "rmayor",
  "Password": "rmayor12344"
 },
 {
  "_id": "S0010",
  "First_Name": "Cruz",
  "Last_Name": "Janie",
  "Email": "cruz.janie@example.com",
  "Username": "cruzjanie",
  "Password": "cjanie45690"
 }
]);
```

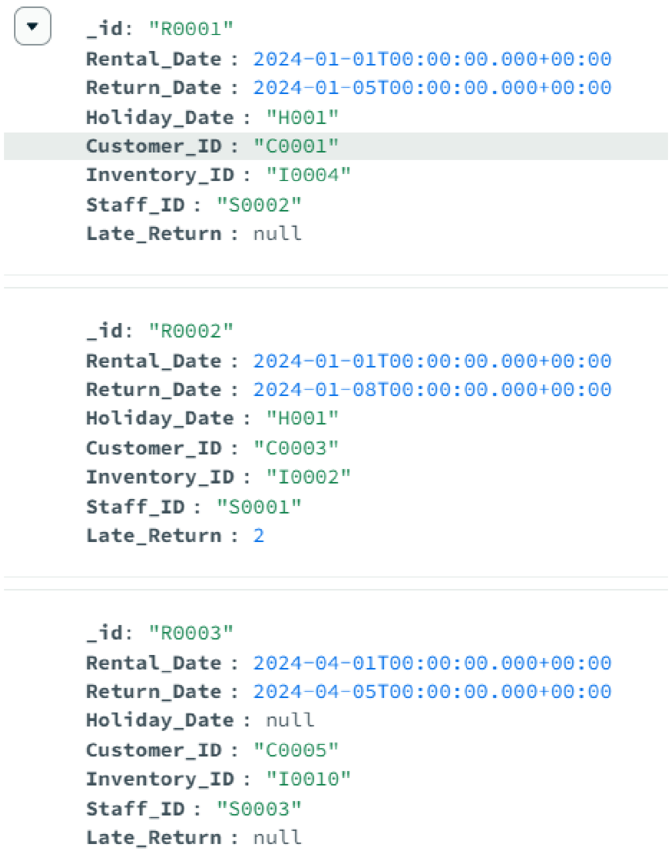

staff

	_id String	First_Name String	Last_Name String	Email String	Username String	Password String
1	"S0001"	"Ahmad"	"Rahman"	"ahmad.rahman@example.co…	"ahmadr"	"password123"
2	"S0002"	"Aisha"	"Ismail"	"aisha.ismail@example.co…	"aishai"	"password456"
3	"S0003"	"Farid"	"Omar"	"farid.omar@example.com"	"farido"	"password789"
4	"S0004"	"Nur"	"Zahra"	"nur.zahra@example.com"	"nurzahra"	"password101"
5	"S0005"	"Hasan"	"Aziz"	"hasan.aziz@example.com"	"hasana"	"password202"
6	"S0006"	"Maya"	"Hassan"	"maya.hassan@example.com"	"mayah"	"password303"
7	"S0007"	"Zain"	"Rahman"	"zain.rahman@example.com"	"zainr"	"password404"
8	"S0008"	"Siti"	"Mariam"	"siti.mariam@example.com"	"sitim"	"password505"
9	"S0009"	"Rivera"	"Mayor"	"rivera.mayor@example.co…	"rmayor"	"rmayor12344"
10	"S0010"	"Cruz"	"Janie"	"cruz.janie@example.com"	"cruzjanie"	"cjanie45690"

Rental collection

```
//function for stock left and calculate the late return day

async function insertRental(rental) {

 try {

  if (!rental.Rental_Date || !rental.Return_Date || !rental.Inventory_ID) {

   throw new Error("Missing required fields: Rental_Date, Return_Date, or Inventory_ID.");

  }


  const rentalDays = Math.ceil(

   (rental.Return_Date - rental.Rental_Date) / (1000 * 60 * 60 * 24)

  );


  rental.Late_Return = rentalDays > 5 ? rentalDays - 5 : null;


  const inventoryItem = await db.Inventory.findOne({ _id: rental.Inventory_ID });

  if (!inventoryItem) {

   throw new Error(`Inventory with ID ${rental.Inventory_ID} not found.`);

  }

  if (inventoryItem.Stock_Left <= 0) {

   throw new Error(`Inventory with ID ${rental.Inventory_ID} is out of stock.`);

  }
```

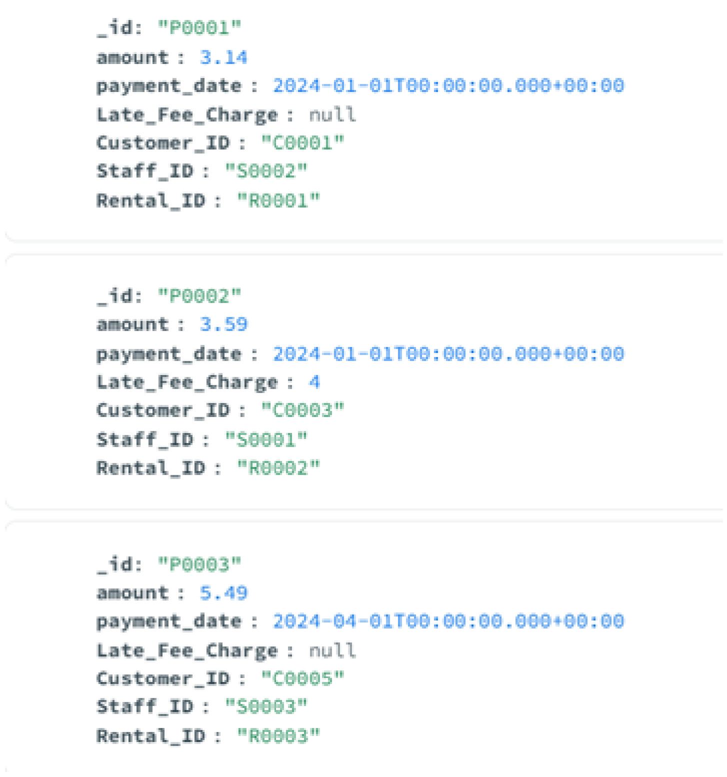
```
    const result = await db.Rental.insertOne(rental);

    if (result.acknowledged) {
      await db.Inventory.updateOne(
        { _id: rental.Inventory_ID },
        { $inc: { Stock_Left: -1 } }
      );
      console.log("Rental inserted and inventory updated successfully!");
    } else {
      console.error("Failed to insert rental.");
    }
  } catch (error) {
    console.error(`Error inserting rental: ${error.message}`);
  }
}
insertRental({
  _id: "R0001",
  Rental_Date: new Date("2024-01-01"),
  Return_Date: new Date("2024-01-05"),
  Holiday_Date: "H001",
  Customer_ID: "C0001",
  Inventory_ID: "I0004",
  Staff_ID: "S0002"
});

insertRental({
  _id: "R0002",
  Rental_Date: new Date("2024-01-01"),
  Return_Date: new Date("2024-01-08"),
  Holiday_Date: "H001",
  Customer_ID: "C0003",
  Inventory_ID: "I0002",
  Staff_ID: "S0001"
});

insertRental({
  _id: "R0003",
  Rental_Date: new Date("2024-04-01"),
  Return_Date: new Date("2024-04-05"),
  Holiday_Date: null,
  Customer_ID: "C0005",
  Inventory_ID: "I0010",
  Staff_ID: "S0003"
});

insertRental({
  _id: "R0004",
  Rental_Date: new Date("2024-05-05"),
  Return_Date: new Date("2024-05-08"),
  Holiday_Date: null,
  Customer_ID: "C0007",
  Inventory_ID: "I0005",
  Staff_ID: "S0004"
});

insertRental({
  _id: "R0005",
  Rental_Date: new Date("2024-07-10"),
  Return_Date: new Date("2024-07-15"),
  Holiday_Date: null,
  Customer_ID: "C0009",
  Inventory_ID: "I0002",
  Staff_ID: "S0001"
});

insertRental({
  _id: "R0006",
  Rental_Date: new Date("2024-08-20"),
  Return_Date: new Date("2024-08-28"),
  Holiday_Date: null,
  Customer_ID: "C0007",
  Inventory_ID: "I0005",
  Staff_ID: "S0005"
});

insertRental({
  _id: "R0007",
  Rental_Date: new Date("2024-09-20"),
  Return_Date: new Date("2024-09-25"),
  Holiday_Date: null,
  Customer_ID: "C0007",
  Inventory_ID: "I0001",
  Staff_ID: "S0006"
});

insertRental({
  _id: "R0008",
  Rental_Date: new Date("2024-09-23"),
  Return_Date: new Date("2024-09-28"),
  Holiday_Date: null,
  Customer_ID: "C0008",
  Inventory_ID: "I0008",
  Staff_ID: "S0008"
});

insertRental({
  _id: "R0009",
  Rental_Date: new Date("2024-10-23"),
  Return_Date: new Date("2024-10-30"),
  Holiday_Date: null,
  Customer_ID: "C0010",
  Inventory_ID: "I0010",
  Staff_ID: "S0007"
});

insertRental({
  _id: "R0010",
  Rental_Date: new Date("2024-11-01"),
  Return_Date: new Date("2024-11-20"),
  Holiday_Date: null,
  Customer_ID: "C0009",
  Inventory_ID: "I0009",
  Staff_ID: "S0008"
});
```



```
_id: "R0001"
Rental_Date : 2024-01-01T00:00:00.000+00:00
Return_Date : 2024-01-05T00:00:00.000+00:00
Holiday_Date : "H001"
Customer_ID : "C0001"
Inventory_ID : "I0004"
Staff_ID : "S0002"
Late_Return : null

_id: "R0002"
Rental_Date : 2024-01-01T00:00:00.000+00:00
Return_Date : 2024-01-08T00:00:00.000+00:00
Holiday_Date : "H001"
Customer_ID : "C0003"
Inventory_ID : "I0002"
Staff_ID : "S0001"
Late_Return : 2

_id: "R0003"
Rental_Date : 2024-04-01T00:00:00.000+00:00
Return_Date : 2024-04-05T00:00:00.000+00:00
Holiday_Date : null
Customer_ID : "C0005"
Inventory_ID : "I0010"
Staff_ID : "S0003"
Late_Return : null
```

Payment collection

```
db.Payment.insertMany([
 {
  "_id": "P0001",
  "amount": 3.14,
  "payment_date": new Date("2024-01-01"),
  "Late_Fee_Charge": null,
  "Customer_ID": "C0001",
  "Staff_ID": "S0002",
  "Rental_ID": "R0001"
 },
 {
  "_id": "P0002",
  "amount": 3.59,
  "payment_date": new Date("2024-01-01"),
  "Late_Fee_Charge": 4,
  "Customer_ID": "C0003",
  "Staff_ID": "S0001",
  "Rental_ID": "R0002"
},
{
  "_id": "P0003",
  "amount": 5.49,
  "payment_date": new Date("2024-04-01"),
  "Late_Fee_Charge": null,
  "Customer_ID": "C0005",
  "Staff_ID": "S0003",
  "Rental_ID": "R0003"
},
{
  "_id": "P0004",
  "amount": 4.49,
  "payment_date": new Date("2024-05-05"),
  "Late_Fee_Charge": null,
  "Customer_ID": "C0007",
  "Staff_ID": "S0004",
  "Rental_ID": "R0004"
},
{
  "_id": "P0005",
  "amount": 3.99,
  "payment_date": new Date("2024-07-10"),
  "Late_Fee_Charge": null,
  "Customer_ID": "C0009",
  "Staff_ID": "S0001",
  "Rental_ID": "R0005"
},
{
  "_id": "P0006",
  "amount": 4.49,
  "payment_date": new Date("2024-08-20"),
  "Late_Fee_Charge": 6,
  "Customer_ID": "C0007",
  "Staff_ID": "S0005",
  "Rental_ID": "R0006"
},
{
  "_id": "P0007",
  "amount": 4.99,
  "payment_date": new Date("2024-09-20"),
  "Late_Fee_Charge": null,
  "Customer_ID": "C0007",
  "Staff_ID": "S0006",
  "Rental_ID": "R0007"
},
{
  "_id": "P0008",
  "amount": 4.99,
  "payment_date": new Date("2024-09-23"),
  "Late_Fee_Charge": null,
  "Customer_ID": "C0008",
    "Staff_ID": "S0008",
    "Rental_ID": "R0008"
  },
  {
    "_id": "P0009",
    "amount": 5.49,
    "payment_date": new Date("2024-10-23"),
    "Late_Fee_Charge": 4,
    "Customer_ID": "C0010",
    "Staff_ID": "S0007",
    "Rental_ID": "R0009"
  },
  {
    "_id": "P0010",
    "amount": 3.99,
    "payment_date": new Date("2024-11-01"),
    "Late_Fee_Charge": 28,
    "Customer_ID": "C0009",
    "Staff_ID": "S0008",
    "Rental_ID": "R0010"
  }
]);
```



```
_id: "P0001"
amount : 3.14
payment_date : 2024-01-01T00:00:00.000+00:00
Late_Fee_Charge : null
Customer_ID : "C0001"
Staff_ID : "S0002"
Rental_ID : "R0001"
```

```
_id: "P0002"
amount : 3.59
payment_date : 2024-01-01T00:00:00.000+00:00
Late_Fee_Charge : 4
Customer_ID : "C0003"
Staff_ID : "S0001"
Rental_ID : "R0002"
```

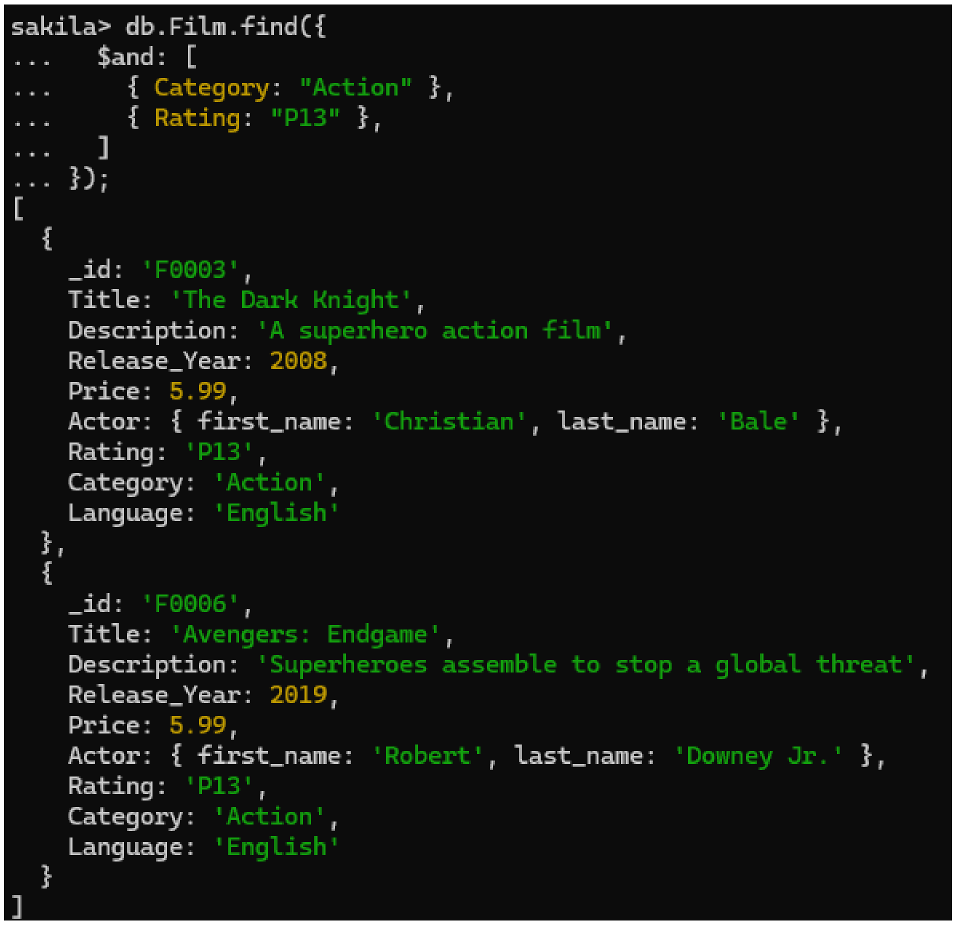
```
_id: "P0003"
amount : 5.49
payment_date : 2024-04-01T00:00:00.000+00:00
Late_Fee_Charge : null
Customer_ID : "C0005"
Staff_ID : "S0003"
Rental_ID : "R0003"
```

5.1 One query with Logical operation (AND, OR, NOT)

Using AND

```
db.Film.find({
 $and: [
  { Category: "Action" },
  { Rating: "P13" },
 ]
});
```

Output:



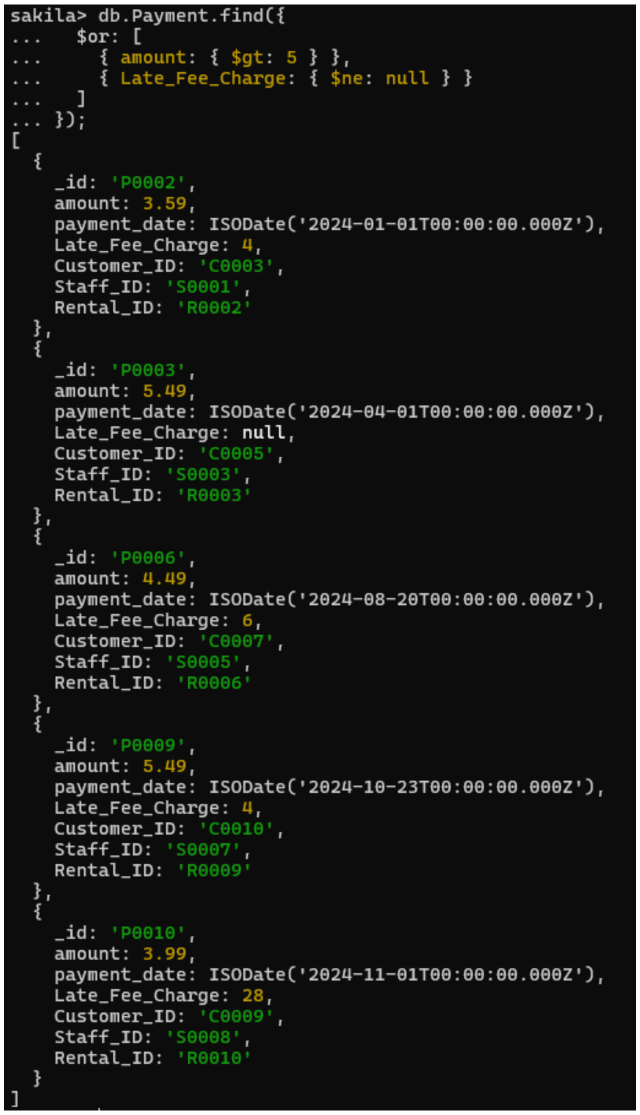
```
sakila> db.Film.find({
...   $and: [
...     { Category: "Action" },
...     { Rating: "P13" },
...   ]
... });
[
  {
    _id: 'F0003',
    Title: 'The Dark Knight',
    Description: 'A superhero action film',
    Release_Year: 2008,
    Price: 5.99,
    Actor: { first_name: 'Christian', last_name: 'Bale' },
    Rating: 'P13',
    Category: 'Action',
    Language: 'English'
  },
  {
    _id: 'F0006',
    Title: 'Avengers: Endgame',
    Description: 'Superheroes assemble to stop a global threat',
    Release_Year: 2019,
    Price: 5.99,
    Actor: { first_name: 'Robert', last_name: 'Downey Jr.' },
    Rating: 'P13',
    Category: 'Action',
    Language: 'English'
  }
]
```

This query is used to find all films where the **Category** is **Action**, and the **Rating** is **P13** to identify action films suitable for teenagers.

Using OR

```
db.Payment.find({
 $or: [
  { amount: { $gt: 5 } },
  { Late_Fee_Charge: { $ne: null } }
 ]
});
```

Output:



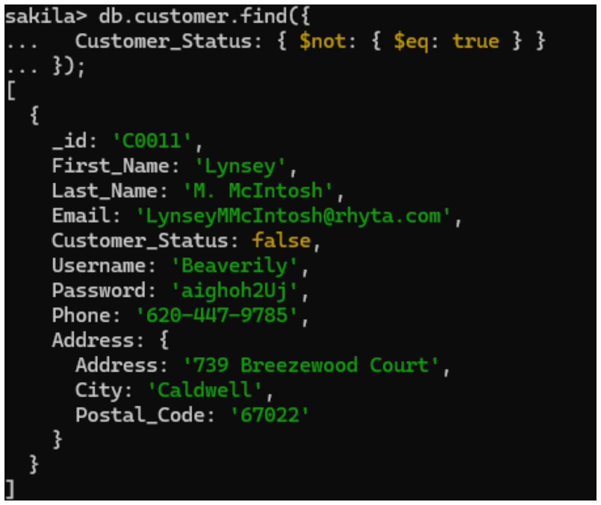
```
sakila> db.Payment.find({
...   $or: [
...     { amount: { $gt: 5 } },
...     { Late_Fee_Charge: { $ne: null } }
...   ]
... });
[
  {
    _id: 'P0002',
    amount: 3.59,
    payment_date: ISODate('2024-01-01T00:00:00.000Z'),
    Late_Fee_Charge: 4,
    Customer_ID: 'C0003',
    Staff_ID: 'S0001',
    Rental_ID: 'R0002'
  },
  {
    _id: 'P0003',
    amount: 5.49,
    payment_date: ISODate('2024-04-01T00:00:00.000Z'),
    Late_Fee_Charge: null,
    Customer_ID: 'C0005',
    Staff_ID: 'S0003',
    Rental_ID: 'R0003'
  },
  {
    _id: 'P0006',
    amount: 4.49,
    payment_date: ISODate('2024-08-20T00:00:00.000Z'),
    Late_Fee_Charge: 6,
    Customer_ID: 'C0007',
    Staff_ID: 'S0005',
    Rental_ID: 'R0006'
  },
  {
    _id: 'P0009',
    amount: 5.49,
    payment_date: ISODate('2024-10-23T00:00:00.000Z'),
    Late_Fee_Charge: 4,
    Customer_ID: 'C0010',
    Staff_ID: 'S0007',
    Rental_ID: 'R0009'
  },
  {
    _id: 'P0010',
    amount: 3.99,
    payment_date: ISODate('2024-11-01T00:00:00.000Z'),
    Late_Fee_Charge: 28,
    Customer_ID: 'C0009',
    Staff_ID: 'S0008',
    Rental_ID: 'R0010'
  }
]
```

This query is used to find the payment which their **amount is greater than 5** *or* their **Late_Fee_Charge is not null**.

Using NOT

```
db.customer.find({
 Customer_Status: { $not: { $eq: true } }
});
```

Output:



```
sakila> db.customer.find({
...   Customer_Status: { $not: { $eq: true } }
... });
[
  {
    _id: 'C0011',
    First_Name: 'Lynsey',
    Last_Name: 'M. McIntosh',
    Email: 'LynseyMMcIntosh@rhyta.com',
    Customer_Status: false,
    Username: 'Beaverily',
    Password: 'aighoh2Uj',
    Phone: '620-447-9785',
    Address: {
      Address: '739 Breezewood Court',
      City: 'Caldwell',
      Postal_Code: '67022'
    }
  }
]
```

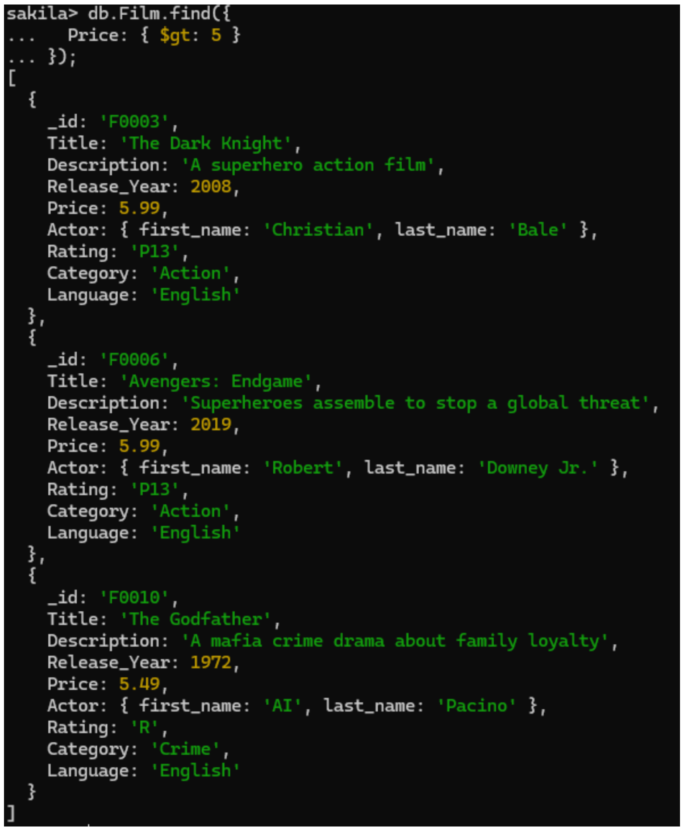
This query is used to find all customer where the **Customer_Status is not true**, which will let us easily to find any inactive customers.

5.2 One query with Comparison operator (greater than, greater than and equal, less than, etc)

Using greater than

```
db.Film.find({
  Price: { $gt: 5 }
});
```

Output:



```
sakila> db.Film.find({
...   Price: { $gt: 5 }
... });
[
  {
    _id: 'F0003',
    Title: 'The Dark Knight',
    Description: 'A superhero action film',
    Release_Year: 2008,
    Price: 5.99,
    Actor: { first_name: 'Christian', last_name: 'Bale' },
    Rating: 'P13',
    Category: 'Action',
    Language: 'English'
  },
  {
    _id: 'F0006',
    Title: 'Avengers: Endgame',
    Description: 'Superheroes assemble to stop a global threat',
    Release_Year: 2019,
    Price: 5.99,
    Actor: { first_name: 'Robert', last_name: 'Downey Jr.' },
    Rating: 'P13',
    Category: 'Action',
    Language: 'English'
  },
  {
    _id: 'F0010',
    Title: 'The Godfather',
    Description: 'A mafia crime drama about family loyalty',
    Release_Year: 1972,
    Price: 5.49,
    Actor: { first_name: 'AI', last_name: 'Pacino' },
    Rating: 'R',
    Category: 'Crime',
    Language: 'English'
  }
]
```

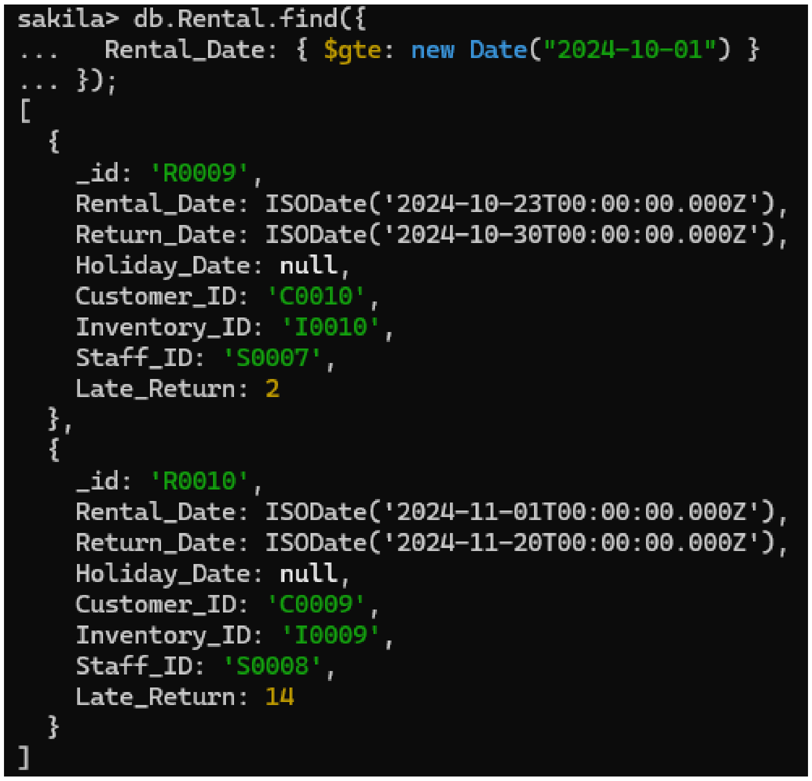
This query is used to find film that has rental price greater than RM5.

$gt is a quey operate that stands for greater than.

Using greater than and equal

```
db.Rental.find({
 Rental_Date: { $gte: new Date("2024-10-01") }
});
```

Output:



```
sakila> db.Rental.find({
...   Rental_Date: { $gte: new Date("2024-10-01") }
... });
[
  {
    _id: 'R0009',
    Rental_Date: ISODate('2024-10-23T00:00:00.000Z'),
    Return_Date: ISODate('2024-10-30T00:00:00.000Z'),
    Holiday_Date: null,
    Customer_ID: 'C0010',
    Inventory_ID: 'I0010',
    Staff_ID: 'S0007',
    Late_Return: 2
  },
  {
    _id: 'R0010',
    Rental_Date: ISODate('2024-11-01T00:00:00.000Z'),
    Return_Date: ISODate('2024-11-20T00:00:00.000Z'),
    Holiday_Date: null,
    Customer_ID: 'C0009',
    Inventory_ID: 'I0009',
    Staff_ID: 'S0008',
    Late_Return: 14
  }
]
```

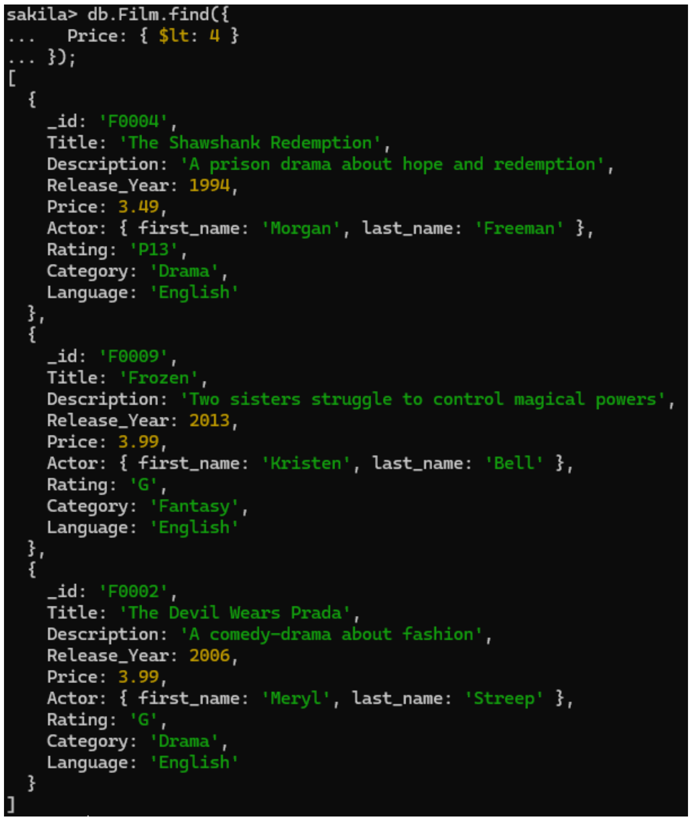
This query is used to find rental on or after 2024-10-01.

$gte is used to get the value which is greater than or equal to another value.

Using less than

```
db.Film.find({
  Price: { $lt: 4 }
});
```

Output:



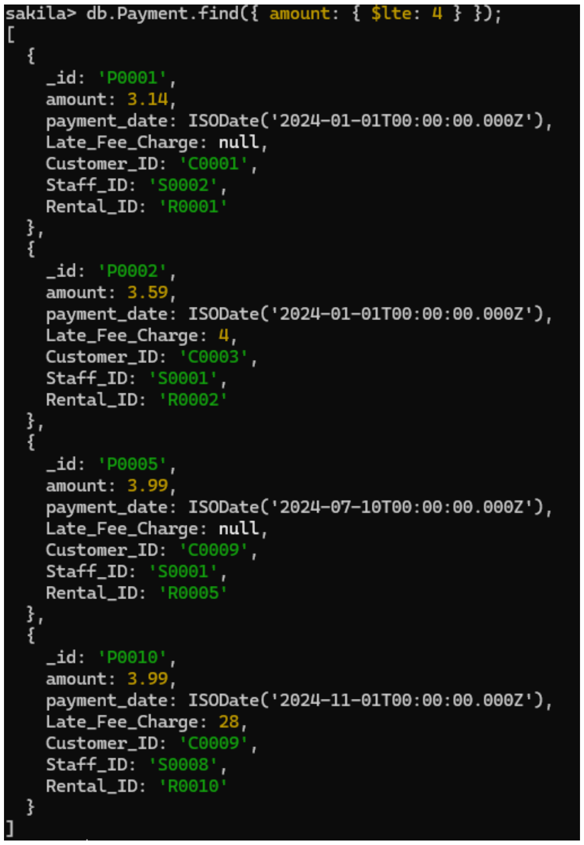
```
sakila> db.Film.find({
...    Price: { $lt: 4 }
... });
[
  {
    _id: 'F0004',
    Title: 'The Shawshank Redemption',
    Description: 'A prison drama about hope and redemption',
    Release_Year: 1994,
    Price: 3.49,
    Actor: { first_name: 'Morgan', last_name: 'Freeman' },
    Rating: 'P13',
    Category: 'Drama',
    Language: 'English'
  },
  {
    _id: 'F0009',
    Title: 'Frozen',
    Description: 'Two sisters struggle to control magical powers',
    Release_Year: 2013,
    Price: 3.99,
    Actor: { first_name: 'Kristen', last_name: 'Bell' },
    Rating: 'G',
    Category: 'Fantasy',
    Language: 'English'
  },
  {
    _id: 'F0002',
    Title: 'The Devil Wears Prada',
    Description: 'A comedy-drama about fashion',
    Release_Year: 2006,
    Price: 3.99,
    Actor: { first_name: 'Meryl', last_name: 'Streep' },
    Rating: 'G',
    Category: 'Drama',
    Language: 'English'
  }
]
```

This query is used to find the film price less than 4 using **$lt**.

Using less than or equal

```
db.Payment.find({
  amount: { $lte: 4 }
});
```

Output:



```
sakila> db.Payment.find({ amount: { $lte: 4 } });
[
  {
    _id: 'P0001',
    amount: 3.14,
    payment_date: ISODate('2024-01-01T00:00:00.000Z'),
    Late_Fee_Charge: null,
    Customer_ID: 'C0001',
    Staff_ID: 'S0002',
    Rental_ID: 'R0001'
  },
  {
    _id: 'P0002',
    amount: 3.59,
    payment_date: ISODate('2024-01-01T00:00:00.000Z'),
    Late_Fee_Charge: 4,
    Customer_ID: 'C0003',
    Staff_ID: 'S0001',
    Rental_ID: 'R0002'
  },
  {
    _id: 'P0005',
    amount: 3.99,
    payment_date: ISODate('2024-07-10T00:00:00.000Z'),
    Late_Fee_Charge: null,
    Customer_ID: 'C0009',
    Staff_ID: 'S0001',
    Rental_ID: 'R0005'
  },
  {
    _id: 'P0010',
    amount: 3.99,
    payment_date: ISODate('2024-11-01T00:00:00.000Z'),
    Late_Fee_Charge: 28,
    Customer_ID: 'C0009',
    Staff_ID: 'S0008',
    Rental_ID: 'R0010'
  }
]
```

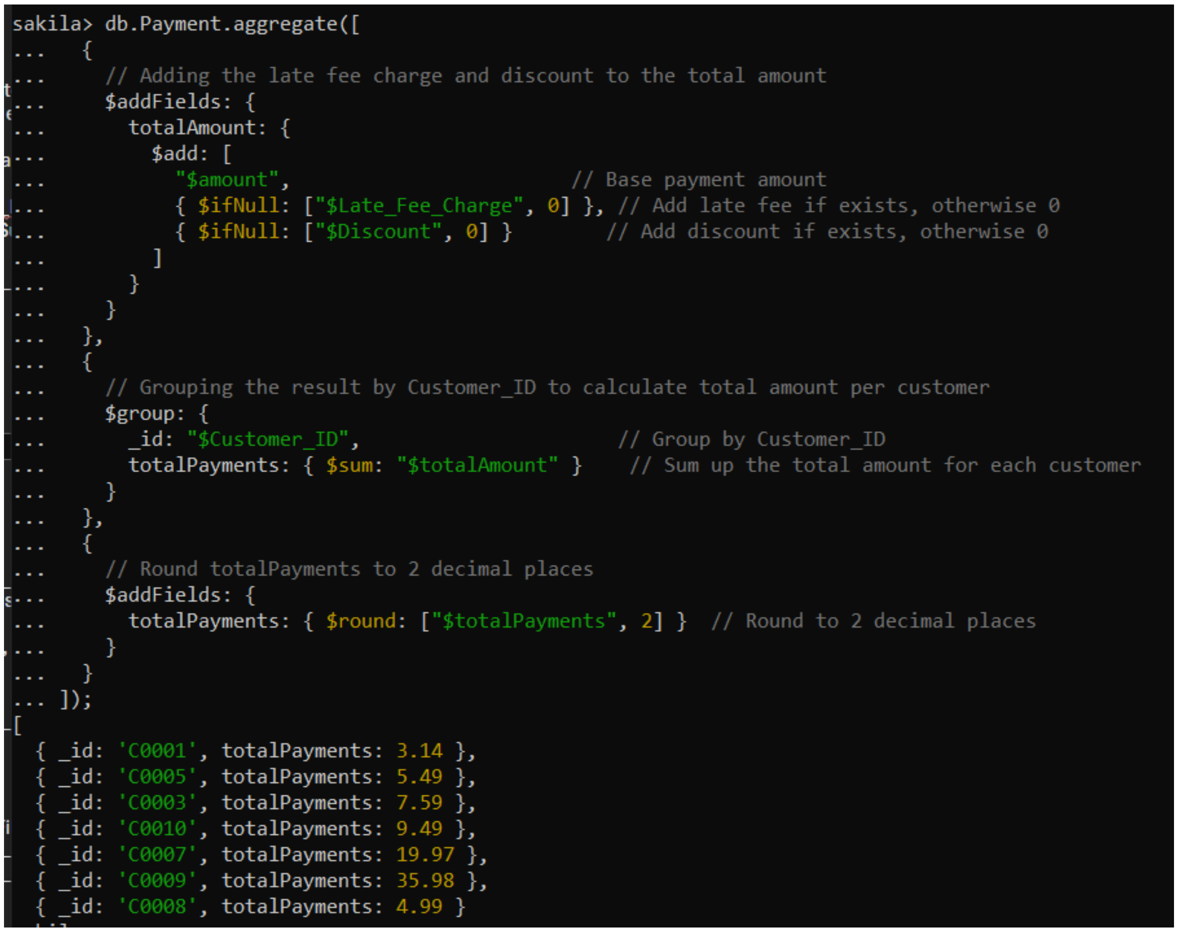
This query is used to find payment amount less than or equal to 4 using **$lte**.

5.3 Aggregate function (sum, average, max, min, count)

Sum

Calculate the total payment amount (including late fees and discounts) for each customer.

```
db.Payment.aggregate([
{
$addFields: {
totalAmount: {
$add: [
"$amount",
{ $ifNull: ["$Late_Fee_Charge", 0] }, /
{ $ifNull: ["$Discount", 0] }
 ] }
 } },
{
 $group: {
_id: "$Customer_ID",
totalPayments: { $sum: "$totalAmount" }
} },
{
 $addFields: {
totalPayments: { $round: ["$totalPayments", 2] }
} } ]);
```



```
sakila> db.Payment.aggregate([
...   {
...     // Adding the late fee charge and discount to the total amount
...     $addFields: {
...       totalAmount: {
...         $add: [
...           "$amount",                        // Base payment amount
...           { $ifNull: ["$Late_Fee_Charge", 0] }, // Add late fee if exists, otherwise 0
...           { $ifNull: ["$Discount", 0] }        // Add discount if exists, otherwise 0
...         ]
...       }
...     }
...   },
...   {
...     // Grouping the result by Customer_ID to calculate total amount per customer
...     $group: {
...       _id: "$Customer_ID",                    // Group by Customer_ID
...       totalPayments: { $sum: "$totalAmount" }    // Sum up the total amount for each customer
...     }
...   },
...   {
...     // Round totalPayments to 2 decimal places
...     $addFields: {
...       totalPayments: { $round: ["$totalPayments", 2] }  // Round to 2 decimal places
...     }
...   }
... ]);
[
  { _id: 'C0001', totalPayments: 3.14 },
  { _id: 'C0005', totalPayments: 5.49 },
  { _id: 'C0003', totalPayments: 7.59 },
  { _id: 'C0010', totalPayments: 9.49 },
  { _id: 'C0007', totalPayments: 19.97 },
  { _id: 'C0009', totalPayments: 35.98 },
  { _id: 'C0008', totalPayments: 4.99 }
```

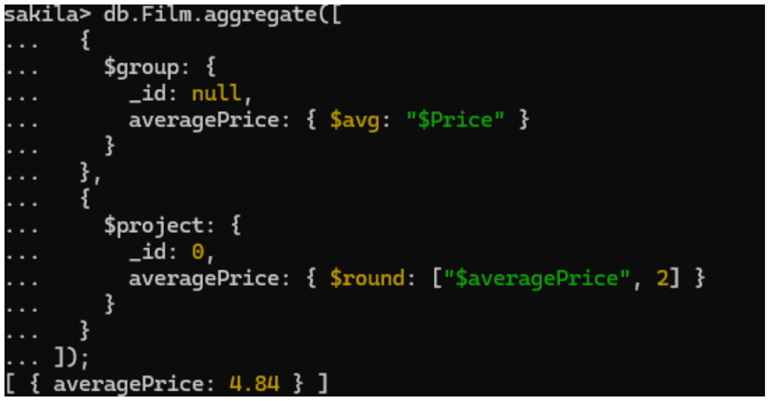
This query provides each **Customer_ID** with their total payment amount rounded to two decimal places, including any late fee charges or discounts.

To calculate the total payment amount per customer, the aggregation pipeline begins by using the **$addFields** stage to create a new field, **totalAmount**. This field is computed by summing the base payment (**amount**), late fee (**Late_Fee_Charge**), and any applicable discounts (**Discount**). To handle cases where the late fee or discount might be **null**, the **$ifNull** operator is used to replace these null values with `0`, ensuring accurate calculations. Next, the **$group** stage organizes the records by **Customer_ID**, summing up the **totalAmount** for each customer to compute their total payments. Finally, the **$addFields** stage is applied again to round the calculated total payments to two decimal places, enhancing readability and presentation of the results.

Average

```
db.Film.aggregate([
 {
  $group: {
   _id: null,
   averagePrice: { $avg: "$Price" }
  }
 },
 {
  $project: {
   _id: 0,
   averagePrice: { $round: ["$averagePrice", 2] }
  }
 }
]);
```

Output:


```
sakila> db.Film.aggregate([
...   {
...     $group: {
...       _id: null,
...       averagePrice: { $avg: "$Price" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       averagePrice: { $round: ["$averagePrice", 2] }
...     }
...   }
... ]);
[ { averagePrice: 4.84 } ]
```

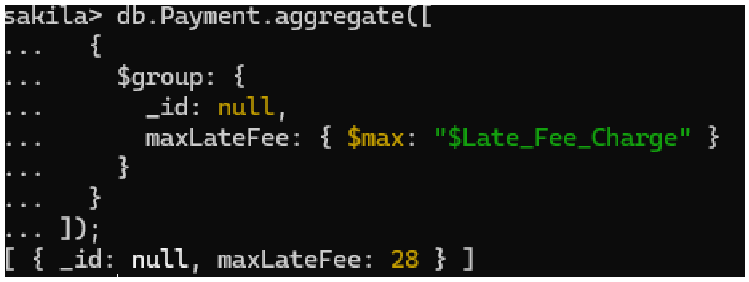
This query is used to find the average film price from film collection.

In **$group** stage**, _id: null** is used so that all documents in the Film collection are grouped together as a single group. Then **averagePrice: { $avg: "$Price" }** calculate the average price from the film collection. The value will then be passed to **$project** stage, **_id: 0** is used to exclude the _id field, then the **$round** is used to round the **averagePrice** to 2 decimal places.

Max

```
db.Payment.aggregate([
 {
   $group: {
     _id: null,
     maxLateFee: { $max: "$Late_Fee_Charge" }
   }
 }
]);
```

Output:


```
sakila> db.Payment.aggregate([
...   {
...     $group: {
...       _id: null,
...       maxLateFee: { $max: "$Late_Fee_Charge" }
...     }
...   }
... ]);
[ { _id: null, maxLateFee: 28 } ]
```

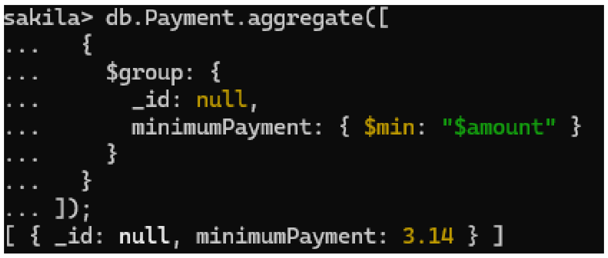
This query is used to find the maximum late fee charge to a customer.

In **$group** stage**, _id: null** is used so that all documents in the Payment collection are grouped together as a single group. Then the **maxLateFee: { $max: "$Late_Fee_Charge" }** is used to find the maximum late fee charge in the payment collection.

Min

```
db.Payment.aggregate([
 {
  $group: {
   _id: null,
   minimumPayment: { $min: "$amount" }
  }
 }
]);
```

Output:



```
sakila> db.Payment.aggregate([
...   {
...     $group: {
...       _id: null,
...       minimumPayment: { $min: "$amount" }
...     }
...   }
... ]);
[ { _id: null, minimumPayment: 3.14 } ]
```

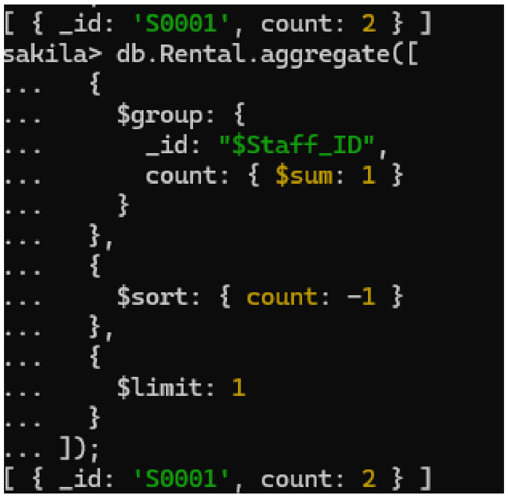
This query will find the payment with minimum payment amount.

In **$group** stage**, _id: null** is used so that all documents in the Payment collection are grouped together as a single group. Then **$min: "$amount"** calculates the smallest value of the amount field.

Count

```
db.Rental.aggregate([
 {
  $group: {
   _id: "$Staff_ID",
   count: { $sum: 1 }
  }
 },
 {
  $sort: { count: -1 }
 },
 {
  $limit: 1
 }
]);
```

Output:



```
[ { _id: 'S0001', count: 2 } ]
sakila> db.Rental.aggregate([
...   {
...     $group: {
...       _id: "$Staff_ID",
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $sort: { count: -1 }
...   },
...   {
...     $limit: 1
...   }
... ]);
[ { _id: 'S0001', count: 2 } ]
```

This query is used to find the staff member who occurs the most in the Rental collection.

In **$group** stage**, _id: "$Staff_ID"** is used to group all documents by the Staff_ID field. Then the **count** field **sums up** the occurrences of each Staff_ID in the Rental collection. In the **$sort** stage, the grouped results by **count** in descending order (**-1**), then the **$limit** limits the output to the top first result only.

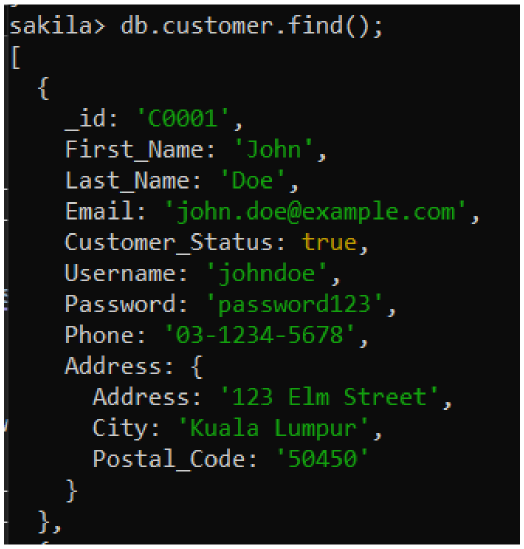
6 Construct the NoSQL command to update particular record(s) based on certain criteria

A.Update a Customer's Phone Number (Using updateOne())

To update a specific customer's phone number, such as changing the Phone detail for the customer with the Username as "johndoe".

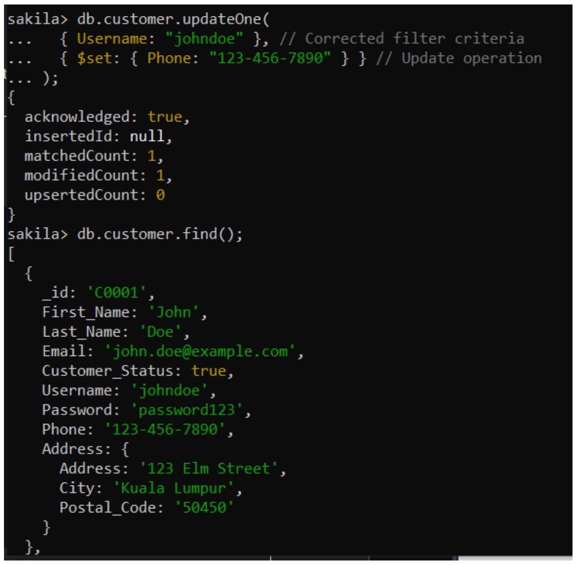
```
db.customer.updateOne(
 { Username: "johndoe" },
 { $set: { Phone: "123-456-7890" } }
);
```

Before:



```
sakila> db.customer.find();
[
  {
    _id: 'C0001',
    First_Name: 'John',
    Last_Name: 'Doe',
    Email: 'john.doe@example.com',
    Customer_Status: true,
    Username: 'johndoe',
    Password: 'password123',
    Phone: '03-1234-5678',
    Address: {
      Address: '123 Elm Street',
      City: 'Kuala Lumpur',
      Postal_Code: '50450'
    }
  },
```

After:



```
sakila> db.customer.updateOne(
...   { Username: "johndoe" }, // Corrected filter criteria
...   { $set: { Phone: "123-456-7890" } } // Update operation
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sakila> db.customer.find();
[
  {
    _id: 'C0001',
    First_Name: 'John',
    Last_Name: 'Doe',
    Email: 'john.doe@example.com',
    Customer_Status: true,
    Username: 'johndoe',
    Password: 'password123',
    Phone: '123-456-7890',
    Address: {
      Address: '123 Elm Street',
      City: 'Kuala Lumpur',
      Postal_Code: '50450'
    }
  },
```

This will update the phone number of the customer with the `Username` "johndoe".

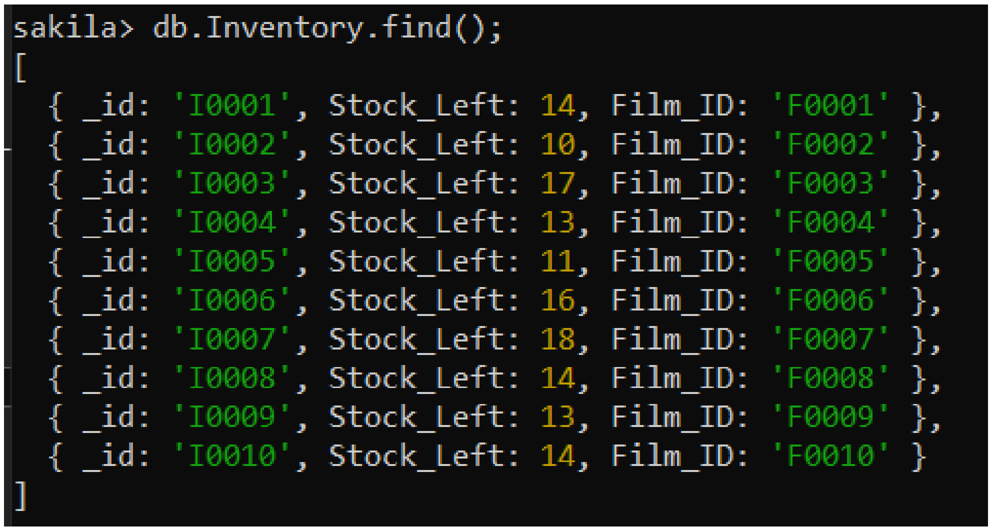
The filter criteria **{ Username: "johndoe" }** is used to specify the customer document that should be updated, targeting the document where the **Username** is **"johndoe"**. This filter identifies the specific document to be modified. The update operation utilizes the **$set** operator, which is used to set the new value for a particular field. In this case, the Phone field is updated to the new phone number **"123-456-7890"**. The **$set** operation only affects the specified field(s) and leaves all other fields (such as `Email`) unchanged. As a result, after the operation, the document for the customer with the **Username "johndoe"** will have the updated phone number, but other fields like **Email** will remain as they were before the update.

B. Update Stock Left for Multiple Inventory Items (Using updateMany())

If the stock level for all inventory items with **Stock_Left** less than 15 needs to be updated to 20.

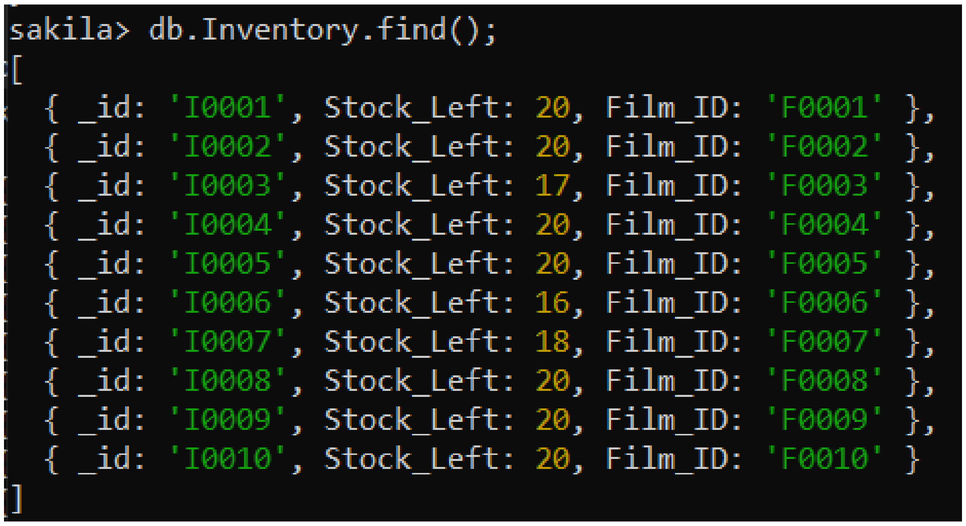
```
db.Inventory.updateMany(

{ Stock_Left: { $lt: 15 } },

{ $set: { Stock_Left: 20 } }

);
```

Before:



```
sakila> db.Inventory.find();
[
  { _id: 'I0001', Stock_Left: 14, Film_ID: 'F0001' },
  { _id: 'I0002', Stock_Left: 10, Film_ID: 'F0002' },
  { _id: 'I0003', Stock_Left: 17, Film_ID: 'F0003' },
  { _id: 'I0004', Stock_Left: 13, Film_ID: 'F0004' },
  { _id: 'I0005', Stock_Left: 11, Film_ID: 'F0005' },
  { _id: 'I0006', Stock_Left: 16, Film_ID: 'F0006' },
  { _id: 'I0007', Stock_Left: 18, Film_ID: 'F0007' },
  { _id: 'I0008', Stock_Left: 14, Film_ID: 'F0008' },
  { _id: 'I0009', Stock_Left: 13, Film_ID: 'F0009' },
  { _id: 'I0010', Stock_Left: 14, Film_ID: 'F0010' }
]
```

After:



```
sakila> db.Inventory.find();
[
  { _id: 'I0001', Stock_Left: 20, Film_ID: 'F0001' },
  { _id: 'I0002', Stock_Left: 20, Film_ID: 'F0002' },
  { _id: 'I0003', Stock_Left: 17, Film_ID: 'F0003' },
  { _id: 'I0004', Stock_Left: 20, Film_ID: 'F0004' },
  { _id: 'I0005', Stock_Left: 20, Film_ID: 'F0005' },
  { _id: 'I0006', Stock_Left: 16, Film_ID: 'F0006' },
  { _id: 'I0007', Stock_Left: 18, Film_ID: 'F0007' },
  { _id: 'I0008', Stock_Left: 20, Film_ID: 'F0008' },
  { _id: 'I0009', Stock_Left: 20, Film_ID: 'F0009' },
  { _id: 'I0010', Stock_Left: 20, Film_ID: 'F0010' }
]
```

This will update the `Stock_Left` for all films with less than 15 in stock to 20.

The filter criteria **{ Stock_Left: { $lt: 15 } }** is used to select all inventory items where the **Stock_Left** is less than 15. The **$lt** operator (less than) ensures that only items with stock levels below 15 are chosen for the update operation. The update operation uses the `$set` operator to set the **Stock_Left** field to 20 for all matching items. The **$set** operator overrides the existing value of **Stock_Left** and updates it to 20. As a result, after the **updateMany()** operation, all inventory items with a **Stock_Left** of less than 15 will have their **Stock_Left** updated to 20, while items with 15 or more stock will remain unaffected and retain their original values.

7 Construct the NoSQL command to delete some specific records based on certain criteria.

A.Delete a single Record (Using deleteOne())

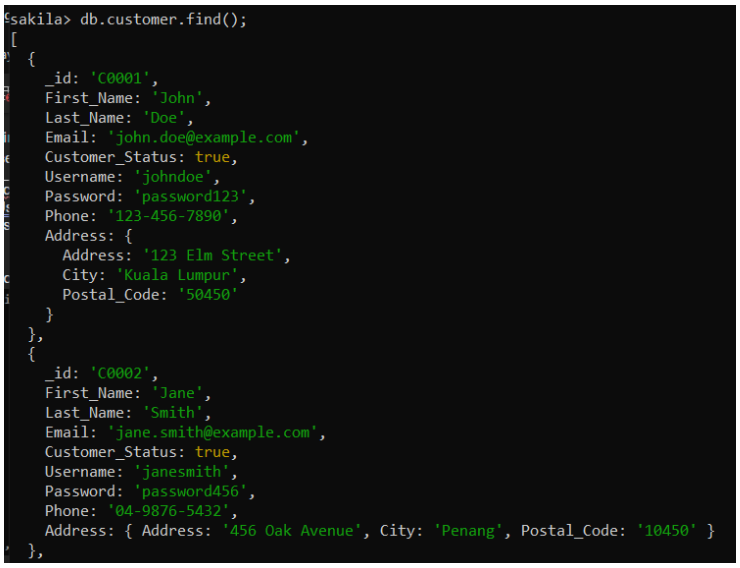
To delete a single record in the `customer` collection where the `Username` is "johndoe"

db.customer.deleteOne(

{ Username: "johndoe" }

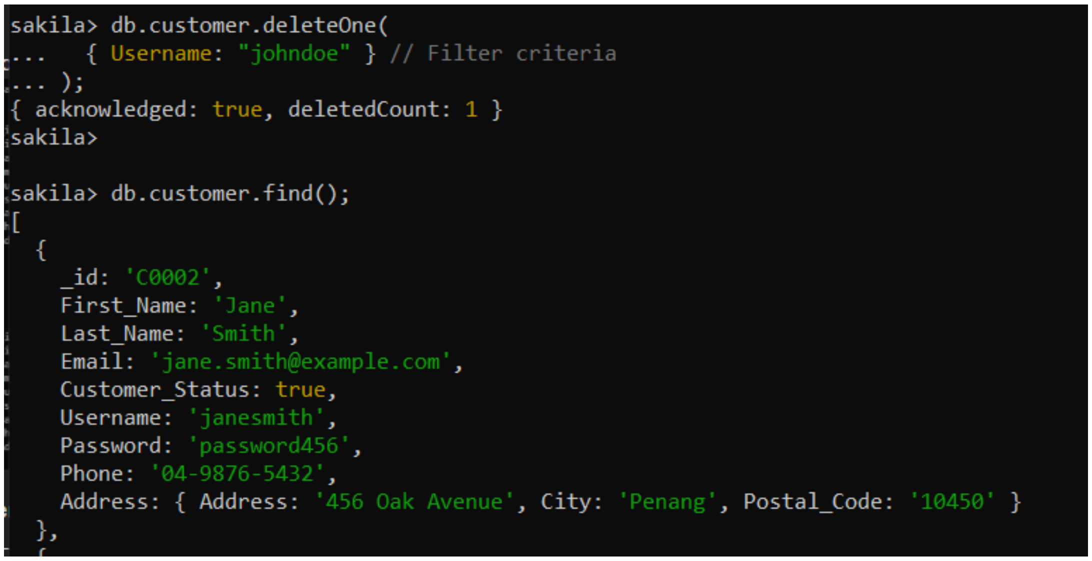
);

Before:



```
sakila> db.customer.find();
[
  {
    _id: 'C0001',
    First_Name: 'John',
    Last_Name: 'Doe',
    Email: 'john.doe@example.com',
    Customer_Status: true,
    Username: 'johndoe',
    Password: 'password123',
    Phone: '123-456-7890',
    Address: {
      Address: '123 Elm Street',
      City: 'Kuala Lumpur',
      Postal_Code: '50450'
    }
  },
  {
    _id: 'C0002',
    First_Name: 'Jane',
    Last_Name: 'Smith',
    Email: 'jane.smith@example.com',
    Customer_Status: true,
    Username: 'janesmith',
    Password: 'password456',
    Phone: '04-9876-5432',
    Address: { Address: '456 Oak Avenue', City: 'Penang', Postal_Code: '10450' }
  },
```

After:



```
sakila> db.customer.deleteOne(
...   { Username: "johndoe" } // Filter criteria
... );
{ acknowledged: true, deletedCount: 1 }
sakila>

sakila> db.customer.find();
[
  {
    _id: 'C0002',
    First_Name: 'Jane',
    Last_Name: 'Smith',
    Email: 'jane.smith@example.com',
    Customer_Status: true,
    Username: 'janesmith',
    Password: 'password456',
    Phone: '04-9876-5432',
    Address: { Address: '456 Oak Avenue', City: 'Penang', Postal_Code: '10450' }
  },
```

This will delete the first matching record with the `Username` "johndoe".

The filter criteria **{ Username: "johndoe" }** specifies the condition used to locate the customer record with the Username field equal to **"johndoe"**. The **deleteOne()** function will search for the first matching document in the collection based on this filter. Once the document is found, it will be deleted. As a result, the record with the Username "johndoe" will no longer exist in the customer collection after the operation. Other records in the collection, if any, will remain unaffected.

B.Delete Multiple Records (deleteMany())

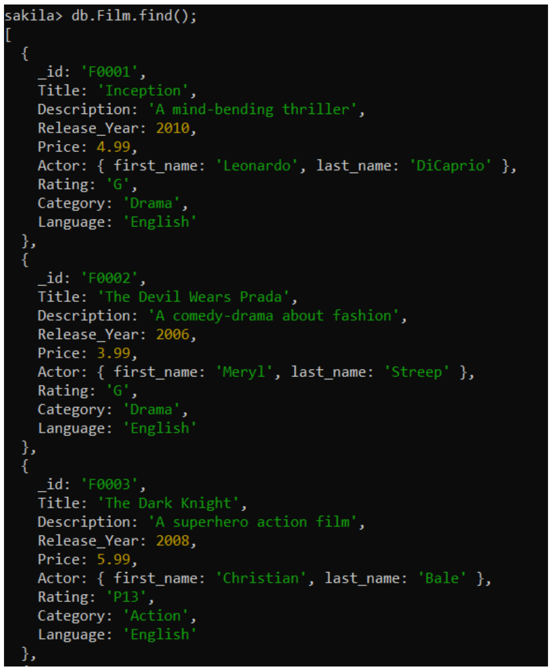
To delete all films in the `Film` collection that were released before the year 2000.

db.Film.deleteMany(

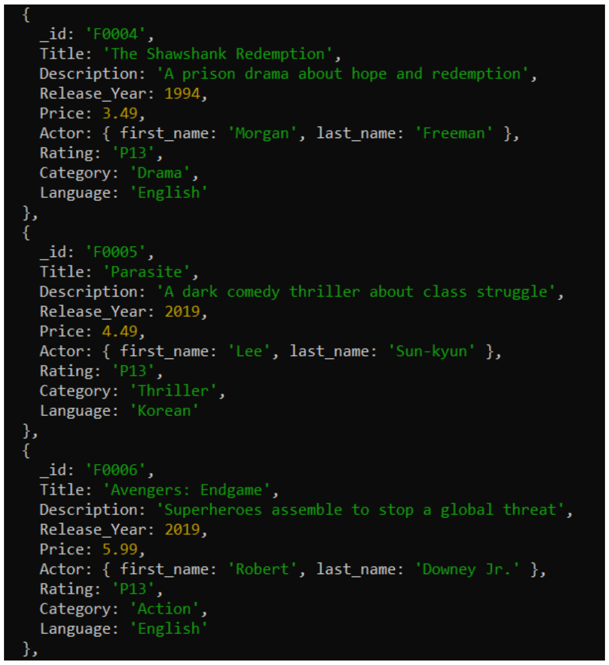
 { Release_Year: { $lt: 2000 } } // Filter criteria (films released before 2000)

);

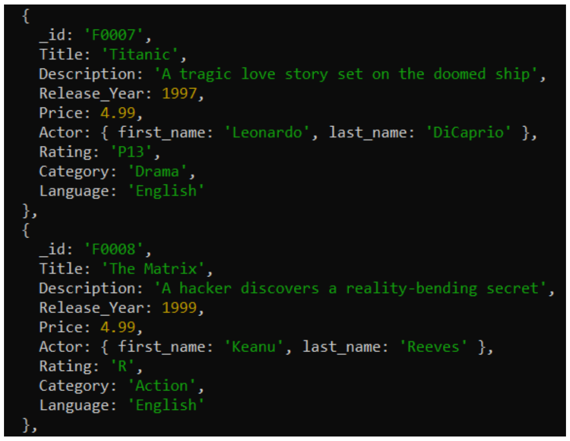
Before:



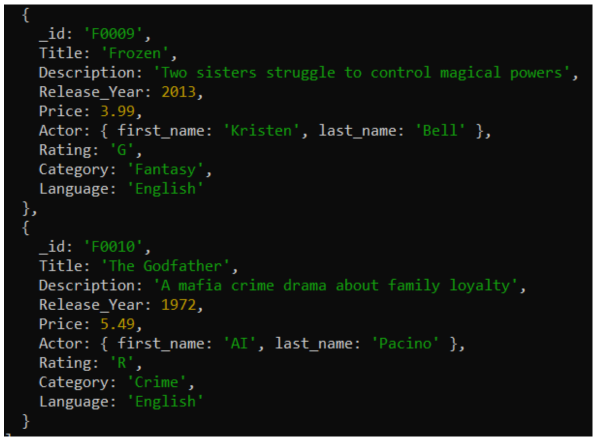
```
sakila> db.Film.find();
[
  {
    _id: 'F0001',
    Title: 'Inception',
    Description: 'A mind-bending thriller',
    Release_Year: 2010,
    Price: 4.99,
    Actor: { first_name: 'Leonardo', last_name: 'DiCaprio' },
    Rating: 'G',
    Category: 'Drama',
    Language: 'English'
  },
  {
    _id: 'F0002',
    Title: 'The Devil Wears Prada',
    Description: 'A comedy-drama about fashion',
    Release_Year: 2006,
    Price: 3.99,
    Actor: { first_name: 'Meryl', last_name: 'Streep' },
    Rating: 'G',
    Category: 'Drama',
    Language: 'English'
  },
  {
    _id: 'F0003',
    Title: 'The Dark Knight',
    Description: 'A superhero action film',
    Release_Year: 2008,
    Price: 5.99,
    Actor: { first_name: 'Christian', last_name: 'Bale' },
    Rating: 'P13',
    Category: 'Action',
    Language: 'English'
  },
```



```
  {
    _id: 'F0004',
    Title: 'The Shawshank Redemption',
    Description: 'A prison drama about hope and redemption',
    Release_Year: 1994,
    Price: 3.49,
    Actor: { first_name: 'Morgan', last_name: 'Freeman' },
    Rating: 'P13',
    Category: 'Drama',
    Language: 'English'
  },
  {
    _id: 'F0005',
    Title: 'Parasite',
    Description: 'A dark comedy thriller about class struggle',
    Release_Year: 2019,
    Price: 4.49,
    Actor: { first_name: 'Lee', last_name: 'Sun-kyun' },
    Rating: 'P13',
    Category: 'Thriller',
    Language: 'Korean'
  },
  {
    _id: 'F0006',
    Title: 'Avengers: Endgame',
    Description: 'Superheroes assemble to stop a global threat',
    Release_Year: 2019,
    Price: 5.99,
    Actor: { first_name: 'Robert', last_name: 'Downey Jr.' },
    Rating: 'P13',
    Category: 'Action',
    Language: 'English'
  },
```

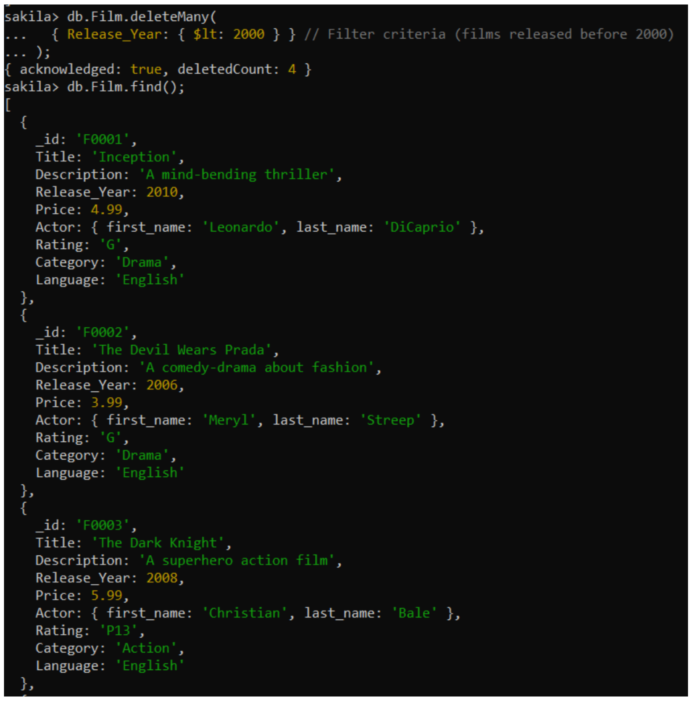


```
  {
    _id: 'F0007',
    Title: 'Titanic',
    Description: 'A tragic love story set on the doomed ship',
    Release_Year: 1997,
    Price: 4.99,
    Actor: { first_name: 'Leonardo', last_name: 'DiCaprio' },
    Rating: 'P13',
    Category: 'Drama',
    Language: 'English'
  },
  {
    _id: 'F0008',
    Title: 'The Matrix',
    Description: 'A hacker discovers a reality-bending secret',
    Release_Year: 1999,
    Price: 4.99,
    Actor: { first_name: 'Keanu', last_name: 'Reeves' },
    Rating: 'R',
    Category: 'Action',
    Language: 'English'
  },
```

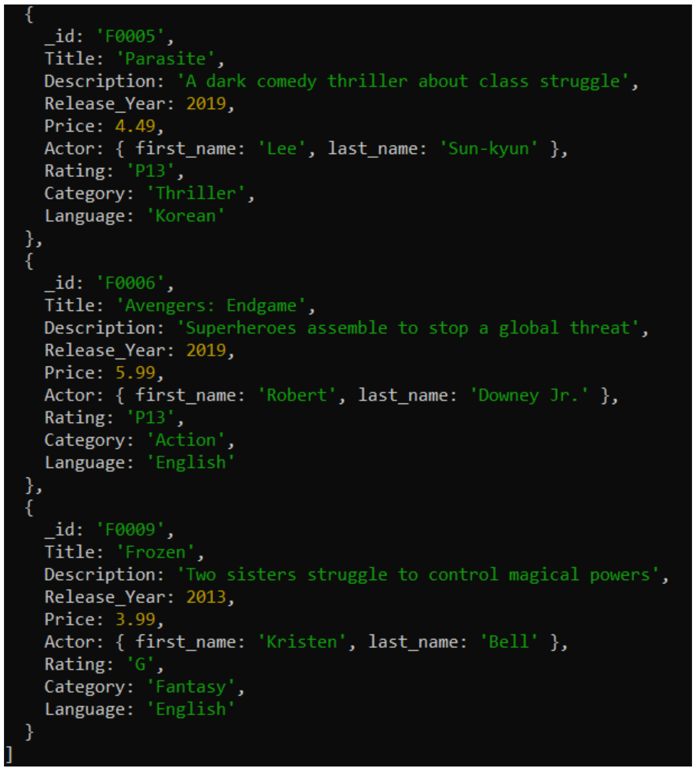


```
  {
    _id: 'F0009',
    Title: 'Frozen',
    Description: 'Two sisters struggle to control magical powers',
    Release_Year: 2013,
    Price: 3.99,
    Actor: { first_name: 'Kristen', last_name: 'Bell' },
    Rating: 'G',
    Category: 'Fantasy',
    Language: 'English'
  },
  {
    _id: 'F0010',
    Title: 'The Godfather',
    Description: 'A mafia crime drama about family loyalty',
    Release_Year: 1972,
    Price: 5.49,
    Actor: { first_name: 'Al', last_name: 'Pacino' },
    Rating: 'R',
    Category: 'Crime',
    Language: 'English'
  }
```

After:



```
sakila> db.Film.deleteMany(
...   { Release_Year: { $lt: 2000 } } // Filter criteria (films released before 2000)
... );
{ acknowledged: true, deletedCount: 4 }
sakila> db.Film.find();
[
  {
    _id: 'F0001',
    Title: 'Inception',
    Description: 'A mind-bending thriller',
    Release_Year: 2010,
    Price: 4.99,
    Actor: { first_name: 'Leonardo', last_name: 'DiCaprio' },
    Rating: 'G',
    Category: 'Drama',
    Language: 'English'
  },
  {
    _id: 'F0002',
    Title: 'The Devil Wears Prada',
    Description: 'A comedy-drama about fashion',
    Release_Year: 2006,
    Price: 3.99,
    Actor: { first_name: 'Meryl', last_name: 'Streep' },
    Rating: 'G',
    Category: 'Drama',
    Language: 'English'
  },
  {
    _id: 'F0003',
    Title: 'The Dark Knight',
    Description: 'A superhero action film',
    Release_Year: 2008,
    Price: 5.99,
    Actor: { first_name: 'Christian', last_name: 'Bale' },
    Rating: 'P13',
    Category: 'Action',
    Language: 'English'
  },
```



```
  {
    _id: 'F0005',
    Title: 'Parasite',
    Description: 'A dark comedy thriller about class struggle',
    Release_Year: 2019,
    Price: 4.49,
    Actor: { first_name: 'Lee', last_name: 'Sun-kyun' },
    Rating: 'P13',
    Category: 'Thriller',
    Language: 'Korean'
  },
  {
    _id: 'F0006',
    Title: 'Avengers: Endgame',
    Description: 'Superheroes assemble to stop a global threat',
    Release_Year: 2019,
    Price: 5.99,
    Actor: { first_name: 'Robert', last_name: 'Downey Jr.' },
    Rating: 'P13',
    Category: 'Action',
    Language: 'English'
  },
  {
    _id: 'F0009',
    Title: 'Frozen',
    Description: 'Two sisters struggle to control magical powers',
    Release_Year: 2013,
    Price: 3.99,
    Actor: { first_name: 'Kristen', last_name: 'Bell' },
    Rating: 'G',
    Category: 'Fantasy',
    Language: 'English'
  }
]
```

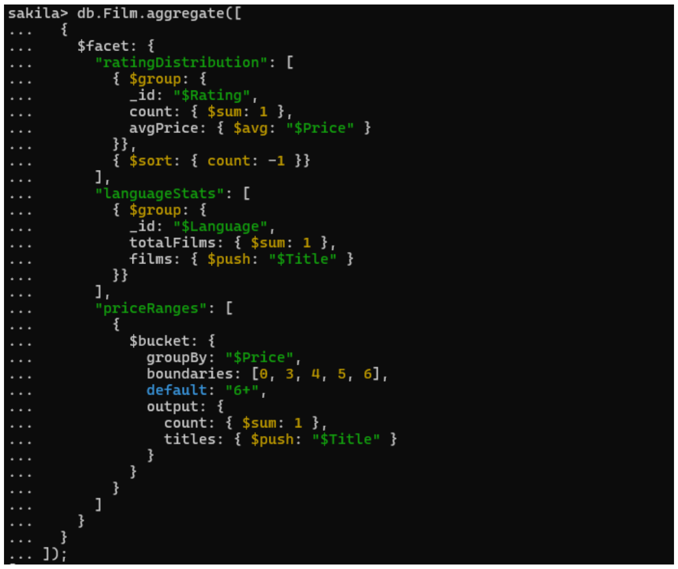
This will delete all films with a `Release_Year` before 2000.

The filter criteria **{ Release_Year: { $lt: 2000 } }** is used to select all films where the Release_Year is **less than 2000**. The **$lt** (less than) operator ensures that only films released before the year 2000 are chosen for deletion. After the **deleteMany()** operation is executed, all films with a Release_Year before 2000 will be deleted from the collection. Any films released in the year 2000 or later will remain intact in the collection, as they do not meet the filter criteria.

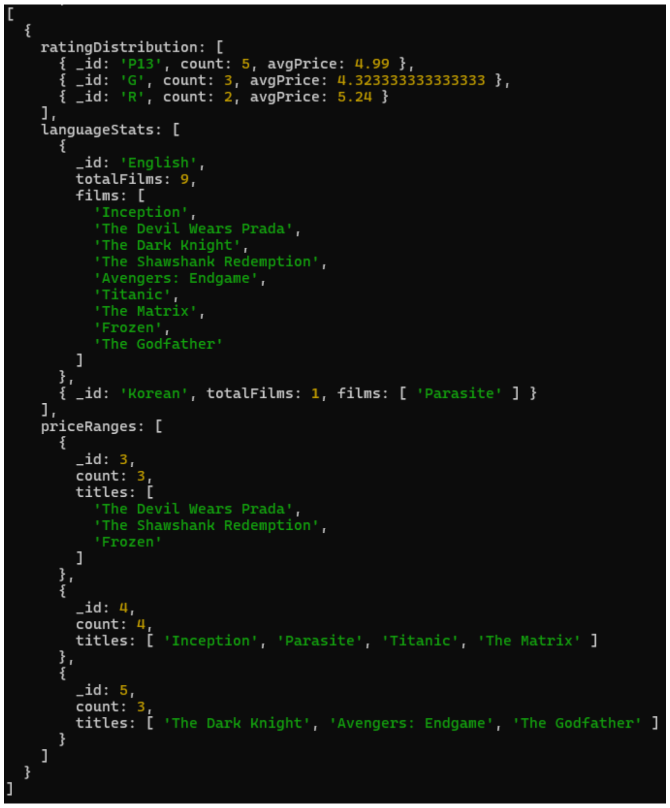
8 Construct any TWO NoSQL commands that are not covered in lecture

A. Faceted Analysis for Film Statistics

```
db.Film.aggregate([
 {
  $facet: {
   "ratingDistribution": [
    { $group: {
      _id: "$Rating",
      count: { $sum: 1 },
      avgPrice: { $avg: "$Price" }
    }},
    { $sort: { count: -1 }}
   ],
   "languageStats": [
    { $group: {
      _id: "$Language",
      totalFilms: { $sum: 1 },
      films: { $push: "$Title" }
    }}
   ],
   "priceRanges": [
    {
      $bucket: {
       groupBy: "$Price",
       boundaries: [0, 3, 4, 5, 6],
      default: "6+",
      output: {
        count: { $sum: 1 },
        titles: { $push: "$Title" }
      }
     }
    }
   ]
  }
 }
]);
```



```
sakila> db.Film.aggregate([
...   {
...     $facet: {
...       "ratingDistribution": [
...         { $group: {
...           _id: "$Rating",
...           count: { $sum: 1 },
...           avgPrice: { $avg: "$Price" }
...         }},
...         { $sort: { count: -1 }}
...       ],
...       "languageStats": [
...         { $group: {
...           _id: "$Language",
...           totalFilms: { $sum: 1 },
...           films: { $push: "$Title" }
...         }}
...       ],
...       "priceRanges": [
...         {
...           $bucket: {
...             groupBy: "$Price",
...             boundaries: [0, 3, 4, 5, 6],
...             default: "6+",
...             output: {
...               count: { $sum: 1 },
...               titles: { $push: "$Title" }
...             }
...           }
...         }
...       ]
...     }
...   }
... ]);
```

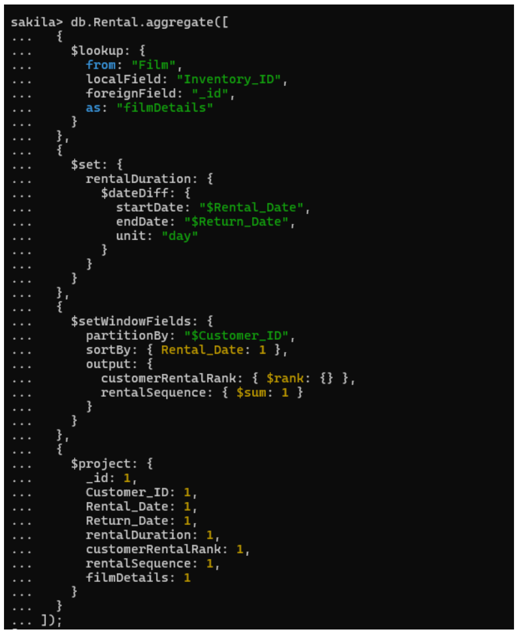


```
[
  {
    ratingDistribution: [
      { _id: 'P13', count: 5, avgPrice: 4.99 },
      { _id: 'G', count: 3, avgPrice: 4.323333333333333 },
      { _id: 'R', count: 2, avgPrice: 5.24 }
    ],
    languageStats: [
      {
        _id: 'English',
        totalFilms: 9,
        films: [
          'Inception',
          'The Devil Wears Prada',
          'The Dark Knight',
          'The Shawshank Redemption',
          'Avengers: Endgame',
          'Titanic',
          'The Matrix',
          'Frozen',
          'The Godfather'
        ]
      },
      { _id: 'Korean', totalFilms: 1, films: [ 'Parasite' ] }
    ],
    priceRanges: [
      {
        _id: 3,
        count: 3,
        titles: [
          'The Devil Wears Prada',
          'The Shawshank Redemption',
          'Frozen'
        ]
      },
      {
        _id: 4,
        count: 4,
        titles: [ 'Inception', 'Parasite', 'Titanic', 'The Matrix' ]
      },
      {
        _id: 5,
        count: 3,
        titles: [ 'The Dark Knight', 'Avengers: Endgame', 'The Godfather' ]
      }
    ]
  }
]
```

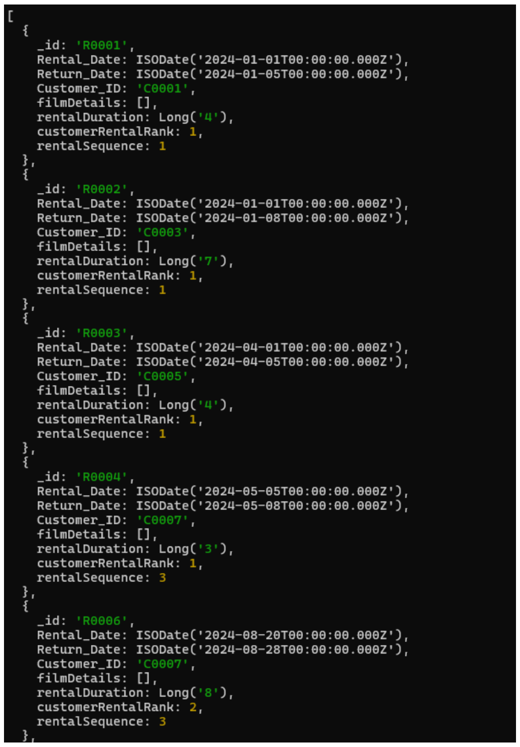
This command utilizes the **$facet** operator to perform multiple aggregation pipelines in a single operation. It simultaneously analyses the film collection from three different perspectives: rating distribution with average prices, language statistics with film titles, and price range categorization. The command offers comprehensive insights into the film inventory by grouping films by rating, calculating average prices, organizing films by language, and categorizing them into price ranges. This multi-dimensional analysis would be useful for business analytics and inventory management.

B. Rental Analysis with Window Functions

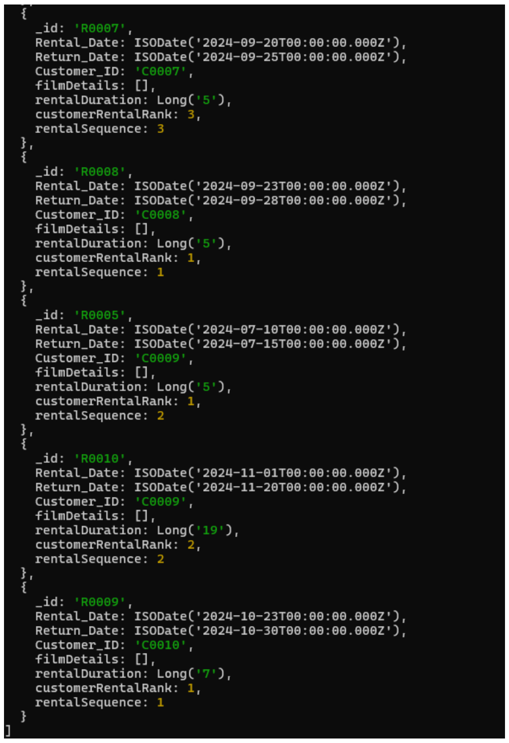
```
db.Rental.aggregate([
 {
  $lookup: {
   from: "Film",
   localField: "Inventory_ID",
   foreignField: "_id",
   as: "filmDetails"
  }
 },
 {
  $set: {
   rentalDuration: {
    $dateDiff: {
     startDate: "$Rental_Date",
     endDate: "$Return_Date",
     unit: "day"
    }
   }
  }
 },
 {
  $setWindowFields: {
   partitionBy: "$Customer_ID",
   sortBy: { Rental_Date: 1 },
   output: {
    customerRentalRank: { $rank: {} },
    rentalSequence: { $sum: 1 }
   }
  }
 },
 {
  $project: {
   _id: 1,
   Customer_ID: 1,
   Rental_Date: 1,
   Return_Date: 1,
   rentalDuration: 1,
   customerRentalRank: 1,
   rentalSequence: 1,
   filmDetails: 1
  }
 }
]);
```



```
sakila> db.Rental.aggregate([
...   {
...     $lookup: {
...       from: "Film",
...       localField: "Inventory_ID",
...       foreignField: "_id",
...       as: "filmDetails"
...     }
...   },
...   {
...     $set: {
...       rentalDuration: {
...         $dateDiff: {
...           startDate: "$Rental_Date",
...           endDate: "$Return_Date",
...           unit: "day"
...         }
...       }
...     }
...   },
...   {
...     $setWindowFields: {
...       partitionBy: "$Customer_ID",
...       sortBy: { Rental_Date: 1 },
...       output: {
...         customerRentalRank: { $rank: {} },
...         rentalSequence: { $sum: 1 }
...       }
...     }
...   },
...   {
...     $project: {
...       _id: 1,
...       Customer_ID: 1,
...       Rental_Date: 1,
...       Return_Date: 1,
...       rentalDuration: 1,
...       customerRentalRank: 1,
...       rentalSequence: 1,
...       filmDetails: 1
...     }
...   }
... ]);
```



```
[
  {
    _id: 'R0001',
    Rental_Date: ISODate('2024-01-01T00:00:00.000Z'),
    Return_Date: ISODate('2024-01-05T00:00:00.000Z'),
    Customer_ID: 'C0001',
    filmDetails: [],
    rentalDuration: Long('4'),
    customerRentalRank: 1,
    rentalSequence: 1
  },
  {
    _id: 'R0002',
    Rental_Date: ISODate('2024-01-01T00:00:00.000Z'),
    Return_Date: ISODate('2024-01-08T00:00:00.000Z'),
    Customer_ID: 'C0003',
    filmDetails: [],
    rentalDuration: Long('7'),
    customerRentalRank: 1,
    rentalSequence: 1
  },
  {
    _id: 'R0003',
    Rental_Date: ISODate('2024-04-01T00:00:00.000Z'),
    Return_Date: ISODate('2024-04-05T00:00:00.000Z'),
    Customer_ID: 'C0005',
    filmDetails: [],
    rentalDuration: Long('4'),
    customerRentalRank: 1,
    rentalSequence: 1
  },
  {
    _id: 'R0004',
    Rental_Date: ISODate('2024-05-05T00:00:00.000Z'),
    Return_Date: ISODate('2024-05-08T00:00:00.000Z'),
    Customer_ID: 'C0007',
    filmDetails: [],
    rentalDuration: Long('3'),
    customerRentalRank: 1,
    rentalSequence: 3
  },
  {
    _id: 'R0006',
    Rental_Date: ISODate('2024-08-20T00:00:00.000Z'),
    Return_Date: ISODate('2024-08-28T00:00:00.000Z'),
    Customer_ID: 'C0007',
    filmDetails: [],
    rentalDuration: Long('8'),
    customerRentalRank: 2,
    rentalSequence: 3
  },
```



```
  {
    _id: 'R0007',
    Rental_Date: ISODate('2024-09-20T00:00:00.000Z'),
    Return_Date: ISODate('2024-09-25T00:00:00.000Z'),
    Customer_ID: 'C0007',
    filmDetails: [],
    rentalDuration: Long('5'),
    customerRentalRank: 3,
    rentalSequence: 3
  },
  {
    _id: 'R0008',
    Rental_Date: ISODate('2024-09-23T00:00:00.000Z'),
    Return_Date: ISODate('2024-09-28T00:00:00.000Z'),
    Customer_ID: 'C0008',
    filmDetails: [],
    rentalDuration: Long('5'),
    customerRentalRank: 1,
    rentalSequence: 1
  },
  {
    _id: 'R0005',
    Rental_Date: ISODate('2024-07-10T00:00:00.000Z'),
    Return_Date: ISODate('2024-07-15T00:00:00.000Z'),
    Customer_ID: 'C0009',
    filmDetails: [],
    rentalDuration: Long('5'),
    customerRentalRank: 1,
    rentalSequence: 2
  },
  {
    _id: 'R0010',
    Rental_Date: ISODate('2024-11-01T00:00:00.000Z'),
    Return_Date: ISODate('2024-11-20T00:00:00.000Z'),
    Customer_ID: 'C0009',
    filmDetails: [],
    rentalDuration: Long('19'),
    customerRentalRank: 2,
    rentalSequence: 2
  },
  {
    _id: 'R0009',
    Rental_Date: ISODate('2024-10-23T00:00:00.000Z'),
    Return_Date: ISODate('2024-10-30T00:00:00.000Z'),
    Customer_ID: 'C0010',
    filmDetails: [],
    rentalDuration: Long('7'),
    customerRentalRank: 1,
    rentalSequence: 1
  }
]
```

This query performs an analysis of rental patterns. It begins by joining rental data with film details using **$lookup**, enhancing the rental information with associated film data. The query then calculates rental durations using **$dateDiff** to understand how long each rental lasted. The most notable aspect is the use of **$setWindowFields**, which enables window function capabilities to analyse customer rental patterns. It ranks rentals chronologically per customer and assigns a sequential number to each rental, providing insights into customer rental frequency and patterns. Finally, **$project** shapes the output to show the most relevant information. This analysis is valuable for understanding customer behaviour, rental durations, and frequency patterns, which can inform inventory management and customer engagement strategies.

References

1. Blog, N. T. (2024, November 17). Recommending for Long-Term member satisfaction at Netflix. Medium. https://netflixtechblog.com/recommending-for-long-term-member-satisfaction-at-netflix-ac15cada49ef
2. Ahmed, A., & Abdulkareem, A. M. (2023). Big data analytics in the entertainment Industry: audience behavior analysis, content recommendation, and Revenue maximization. *Reviews of Contemporary Business Analytics*, 6(1), 88-102.

3. Awan, M. J., Khan, R. A., Nobanee, H., Yasin, A., Anwar, S. M., Naseem, U., & Singh, V. P. (2021). A recommendation engine for predicting movie ratings using a big data approach. Electronics, 10(10), 1215.
4. Bharadiya, J. P. (2023). A comparative study of business intelligence and artificial intelligence with big data analytics. American Journal of Artificial Intelligence, 7(1), 24.