Topics

1. Review
2. Adding and Removing Documents
3. Searching for Documents

PL/SQL Review

- What are the two types of cursors:
- Implicit cursors
- Explicit cursors
- What is an implicit cursor?
- The implicit cursors are automatically created while a statement such as INSERT, UPDATE, DELETE, SELECT and SELECT INTO are executed.
- The implicit cursor attributes can be used to determine if any rows have been affected as a result of the execution of a SQL statement.

PL/SQL Review

- For an implicit cursor, what returns the number of rows affected by the execution of a DML statement or a SELECT statement
- SQL%ROWCOUNT

PL/SQL Review

- How are %Type and %RowType used
- you do not have to know the definition of a column in a table when setting up your variable or the definition of an entire row of columns when setting up multiple variables.
- if the type definition for that column or a row of column changes, your code does not have to be updated – it will automatically pick up the new type definition

PL/SQL Review

- What are five steps involved when using an explicit cursor
- DECLARE the cursor
- OPEN the cursor
- FETCH rows from the cursor
- Do something with the fetched row
- CLOSE the cursor

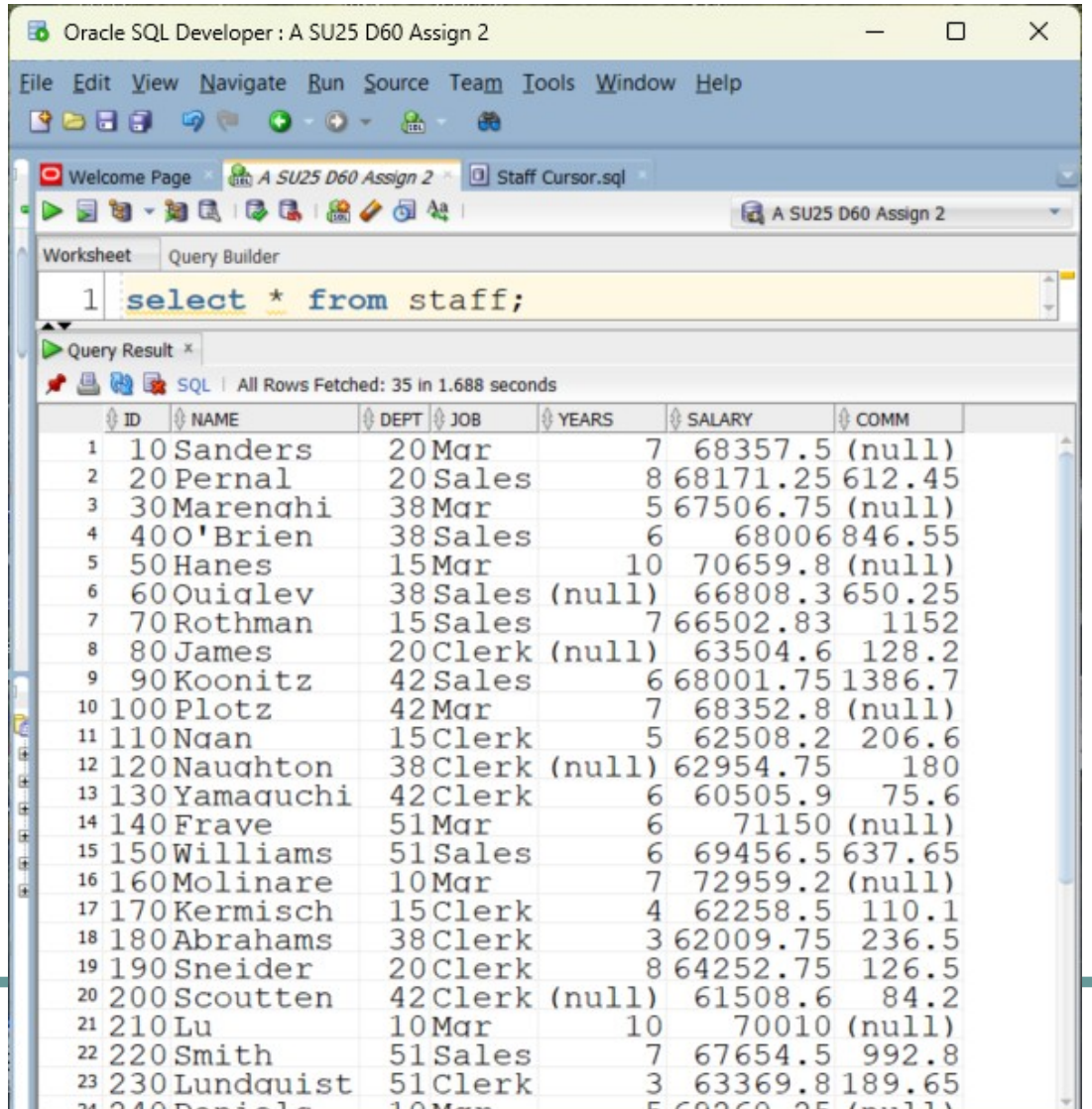
PL/SQL Review

- How do you Define a Cursor?
- **CURSOR** cursor_name **IS** select_statement;
- OPEN cursor_name;
- If you want to include parameters?
- **CURSOR** cursor_name (parm1 parm1type,parm2 parm2type)
- **IS** select_statement;
- OPEN cursor_name(parm1value, parm2value);
- How do you retrieve a row from your cursor?
- Fetch cursor_name into variable list

PL/SQL Review

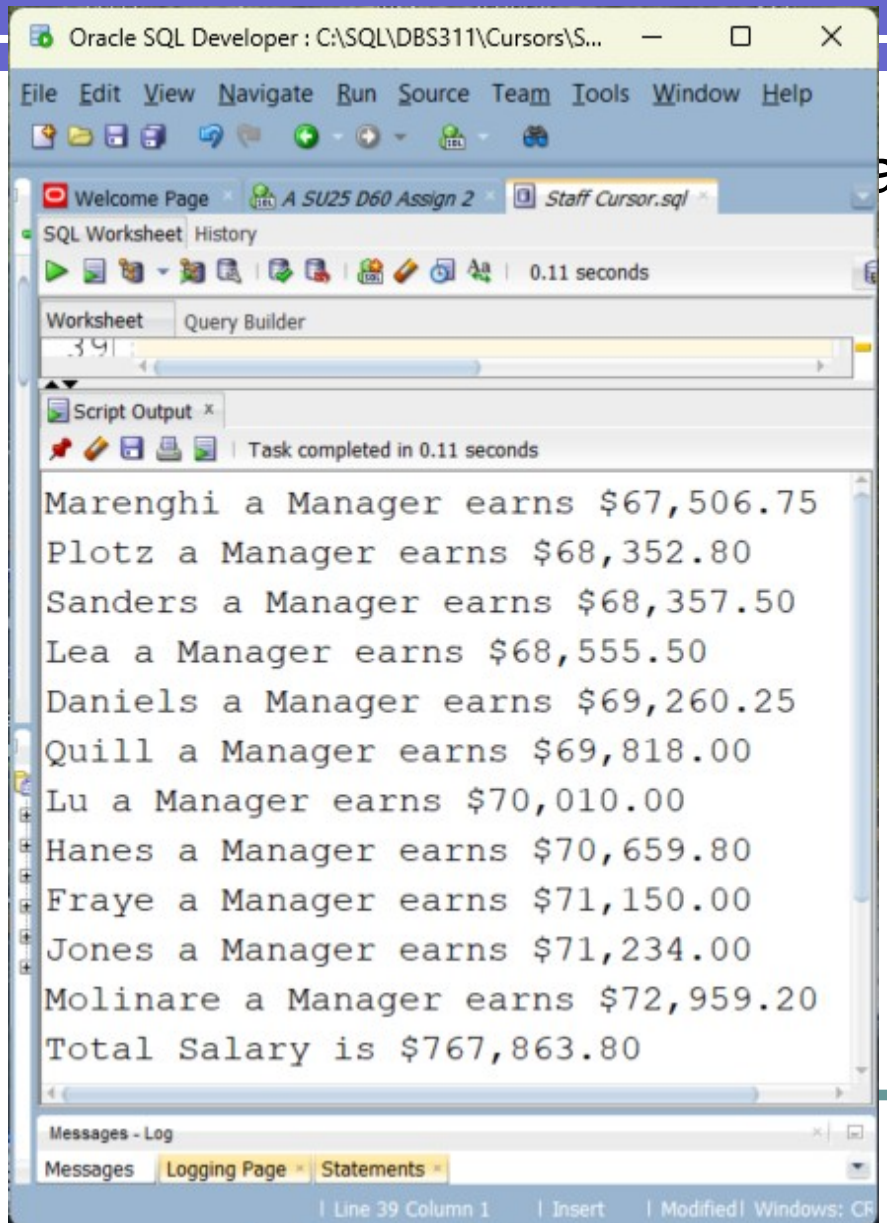
- How do you employ a cursor without explicitly stating the open, close and fetch?
- For VQ in cursor_name
- LOOP
- DBMS_OUTPUT.PUT_LINE(vq.col1, vq.col2)
- END LOOP;

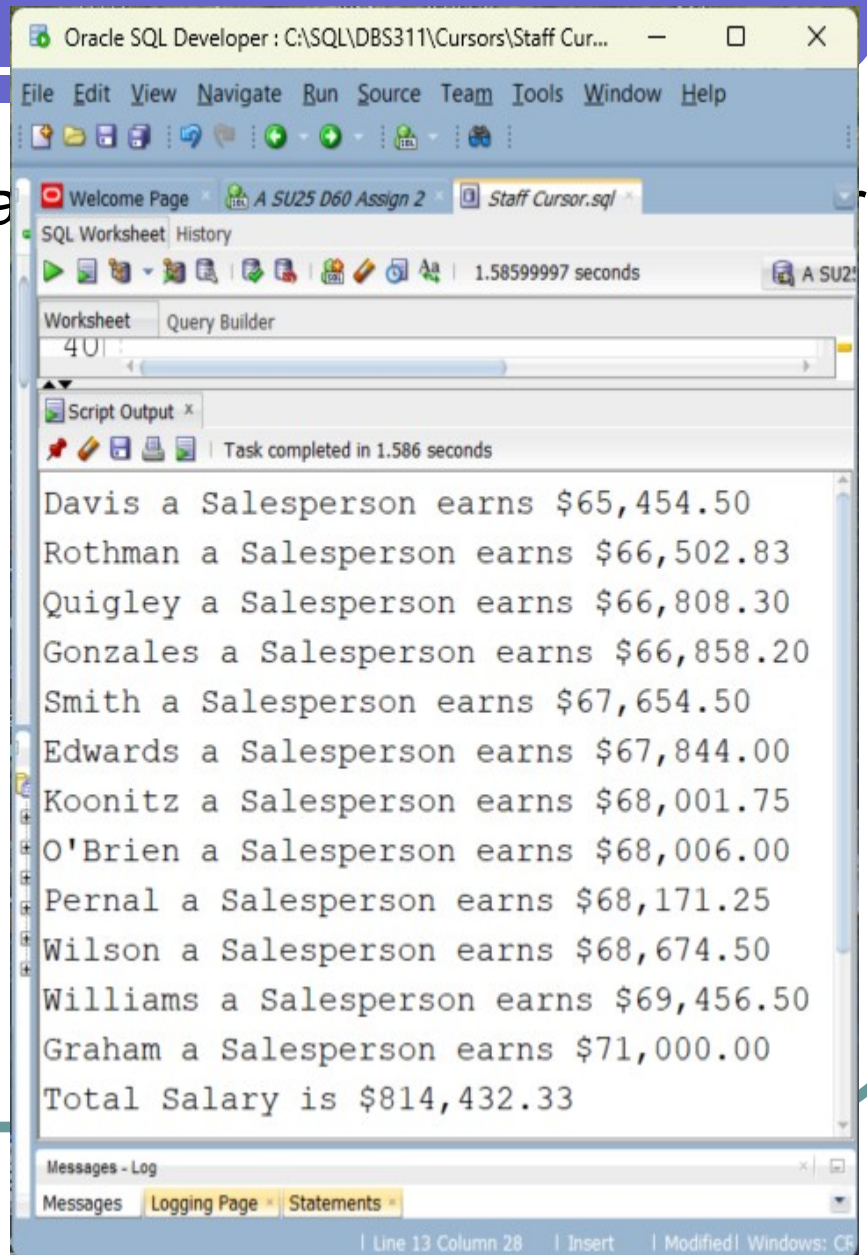
STAFF



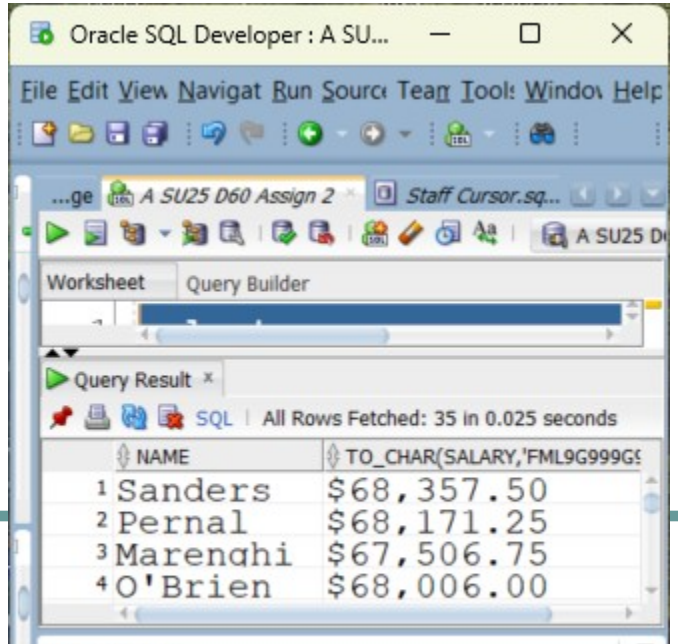
	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	10	Sanders	20	Mgr	7	68357.5	(null)
2	20	Pernal	20	Sales	8	68171.25	612.45
3	30	Marenghi	38	Mgr	5	67506.75	(null)
4	40	O'Brien	38	Sales	6	68006	846.55
5	50	Hanes	15	Mgr	10	70659.8	(null)
6	60	Quigley	38	Sales	(null)	66808.3	650.25
7	70	Rothman	15	Sales	7	66502.83	1152
8	80	James	20	Clerk	(null)	63504.6	128.2
9	90	Koonitz	42	Sales	6	68001.75	1386.7
10	100	Plotz	42	Mgr	7	68352.8	(null)
11	110	Ngan	15	Clerk	5	62508.2	206.6
12	120	Naughton	38	Clerk	(null)	62954.75	180
13	130	Yamaguchi	42	Clerk	6	60505.9	75.6
14	140	Fraye	51	Mgr	6	71150	(null)
15	150	Williams	51	Sales	6	69456.5	637.65
16	160	Molinare	10	Mgr	7	72959.2	(null)
17	170	Kermisch	15	Clerk	4	62258.5	110.1
18	180	Abrahams	38	Clerk	3	62009.75	236.5
19	190	Sneider	20	Clerk	8	64252.75	126.5
20	200	Scoutten	42	Clerk	(null)	61508.6	84.2
21	210	Lu	10	Mgr	10	70010	(null)
22	220	Smith	51	Sales	7	67654.5	992.8
23	230	Lundquist	51	Clerk	3	63369.8	189.65

Explicit Cursors in an PL/SQL Review exercise





- Formatting the salary
- select name,
- to_char(salary,'fmL9G999G999D00')
- From staff;



Explicit Cursors in an PL/SQL Review exercise

- Solution taken up in class.

Review

- What are the differences between a User Defined Function and a Stored Procedure
- UDF's always return a value, returning a value is optional with a stored procedure
- Functions can be used in SQL statements while procedures can't
- Functions are usually used for computations while procedures are used for executing business logic

Review

- What are the differences between a User Defined Function and a Stored Procedure
- UDF's always return a value, returning a value is optional with a stored procedure
- Functions can be used in SQL statements while procedures can't
- Functions are usually used for computations while procedures are used for executing business logic

Review

- Where can user defined functions appear in SQL statements?
- In the column list, where, order by and group by clauses of a Select statement
- In the Where clause of an Update statement
- In the Where clause of a Delete statement
- In the Values clause of an Insert statement
- Can a user defined function be set up as UDF() – no value passed to it.

MongoDB

- What does NoSQL stand for
- Not only SQL
- MongoDB uses a NoSQL database approach, how would you compare it to what you have been doing with Oracle and DB2
- MongoDB is not a relational database management system
- MongoDB is document orientated
- How is data stored with MongoDB
- Data is stored with Key value pairs
- What are the strengths with MongoDB
- It does not have a predefined schema
- Scales well
- Does not require complicated joins over several tables

Review

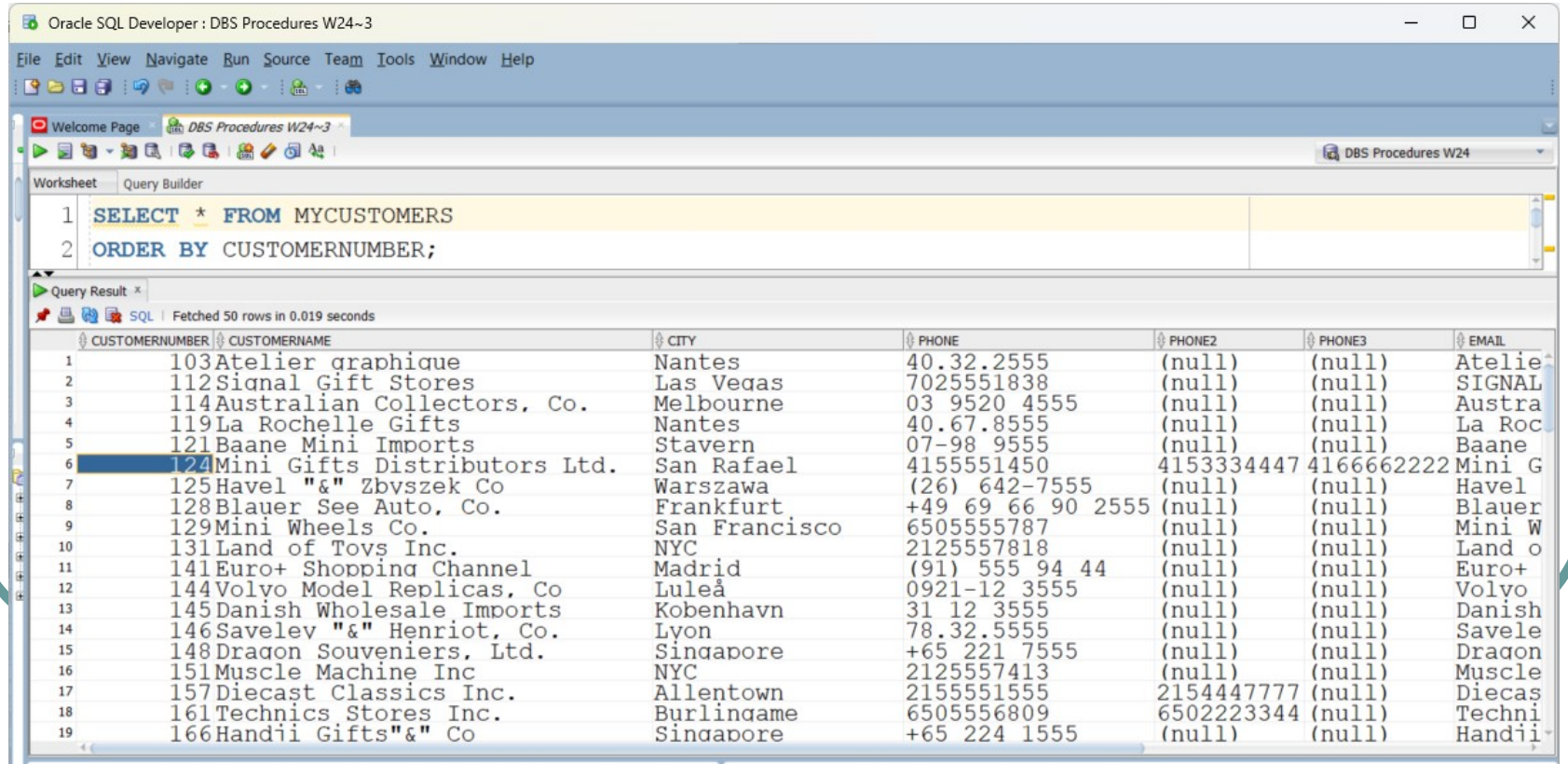
- If you were comparing MongoDB and a realational database, how would you describe a document?
- Similar to a row in a RDBMS
- What would you compare a RDBMS table to?
- With MongoDB that would be a collection
- With MongoDB, a group of collections would be
- A Database

Review

- What are MongDB's three reserved database names?
- Admin
- This database is the root database.
- Users added to the admin database have all permissions to all databases.
- Only admin users are authorized to run certain server commands.
- Local
- This database stores any local connections on a single server.
- Config
- A config server stores the clusters' metadata.

Why MongoDB

- Multiple phone numbers for some customers would not be handled this way in a Relational DBMS



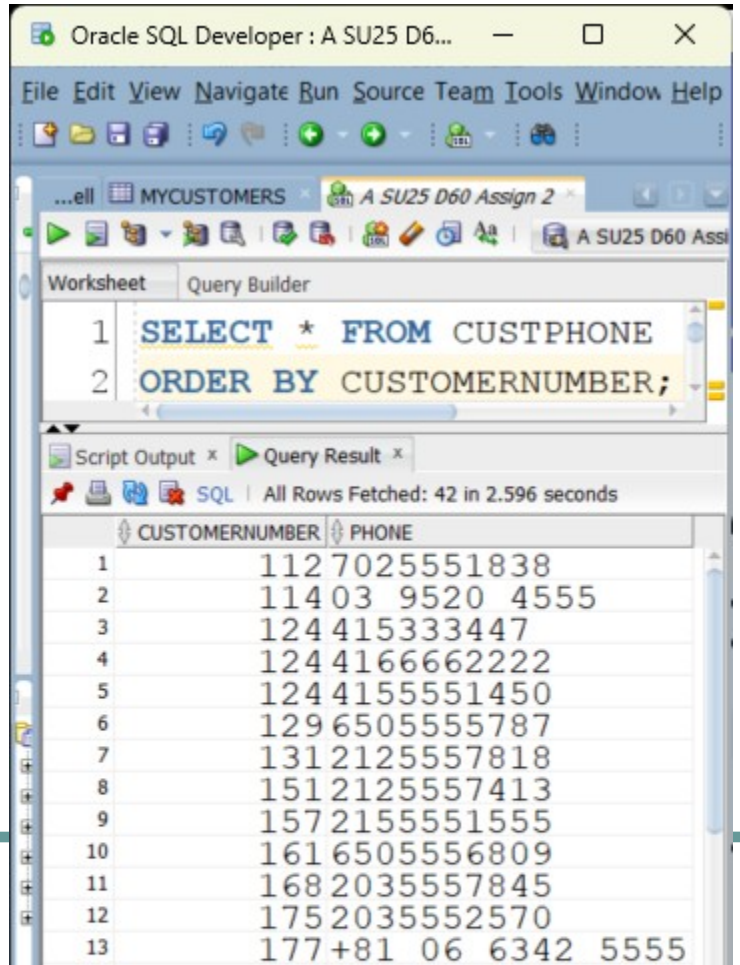
	CUSTOMERNUMBER	CUSTOMERNAME	CITY	PHONE	PHONE2	PHONE3	EMAIL
1	103	Atelier graphique	Nantes	40.32.2555	(null)	(null)	Atelie
2	112	Signal Gift Stores	Las Vegas	7025551838	(null)	(null)	SIGNAL
3	114	Australian Collectors, Co.	Melbourne	03 9520 4555	(null)	(null)	Austra
4	119	La Rochelle Gifts	Nantes	40.67.8555	(null)	(null)	La Roc
5	121	Baane Mini Imports	Stavern	07-98 9555	(null)	(null)	Baane
6	124	Mini Gifts Distributors Ltd.	San Rafael	4155551450	4153334447	4166662222	Mini G
7	125	Havel "&" Zbyszek Co	Warszawa	(26) 642-7555	(null)	(null)	Havel
8	128	Blauer See Auto, Co.	Frankfurt	+49 69 66 90 2555	(null)	(null)	Blauer
9	129	Mini Wheels Co.	San Francisco	6505555787	(null)	(null)	Mini W
10	131	Land of Toys Inc.	NYC	2125557818	(null)	(null)	Land o
11	141	Euro+ Shopping Channel	Madrid	(91) 555 94 44	(null)	(null)	Euro+
12	144	Volvo Model Replicas, Co	Luleå	0921-12 3555	(null)	(null)	Volvo
13	145	Danish Wholesale Imports	Kobenhavn	31 12 3555	(null)	(null)	Danish
14	146	Saveley "&" Henriot, Co.	Lyon	78.32.5555	(null)	(null)	Savele
15	148	Dragon Souveniers, Ltd.	Singapore	+65 221 7555	(null)	(null)	Dragon
16	151	Muscle Machine Inc	NYC	2125557413	(null)	(null)	Muscle
17	157	Diecast Classics Inc.	Allentown	2155551555	2154447777	(null)	Diecas
18	161	Technics Stores Inc.	Burlingame	6505556809	6502223344	(null)	Techni
19	166	Handji Gifts"&" Co	Singapore	+65 224 1555	(null)	(null)	Handji

Why MongoDB

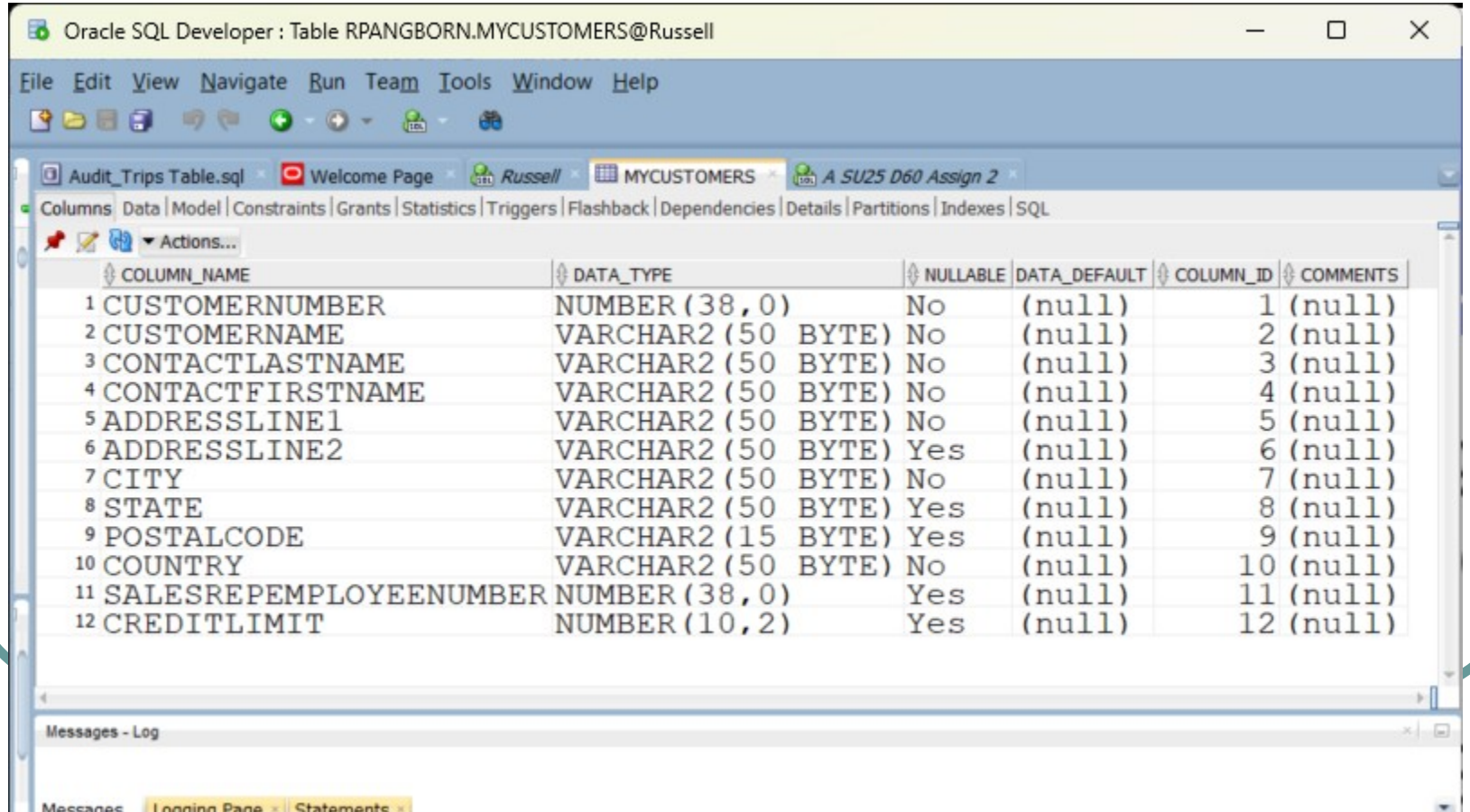
- In Relational DBMS phone table phone information would be split into a separate table for handling customers with multiple phone numbers

RDBMS JOIN

- Create a separate table for Phone Numbers

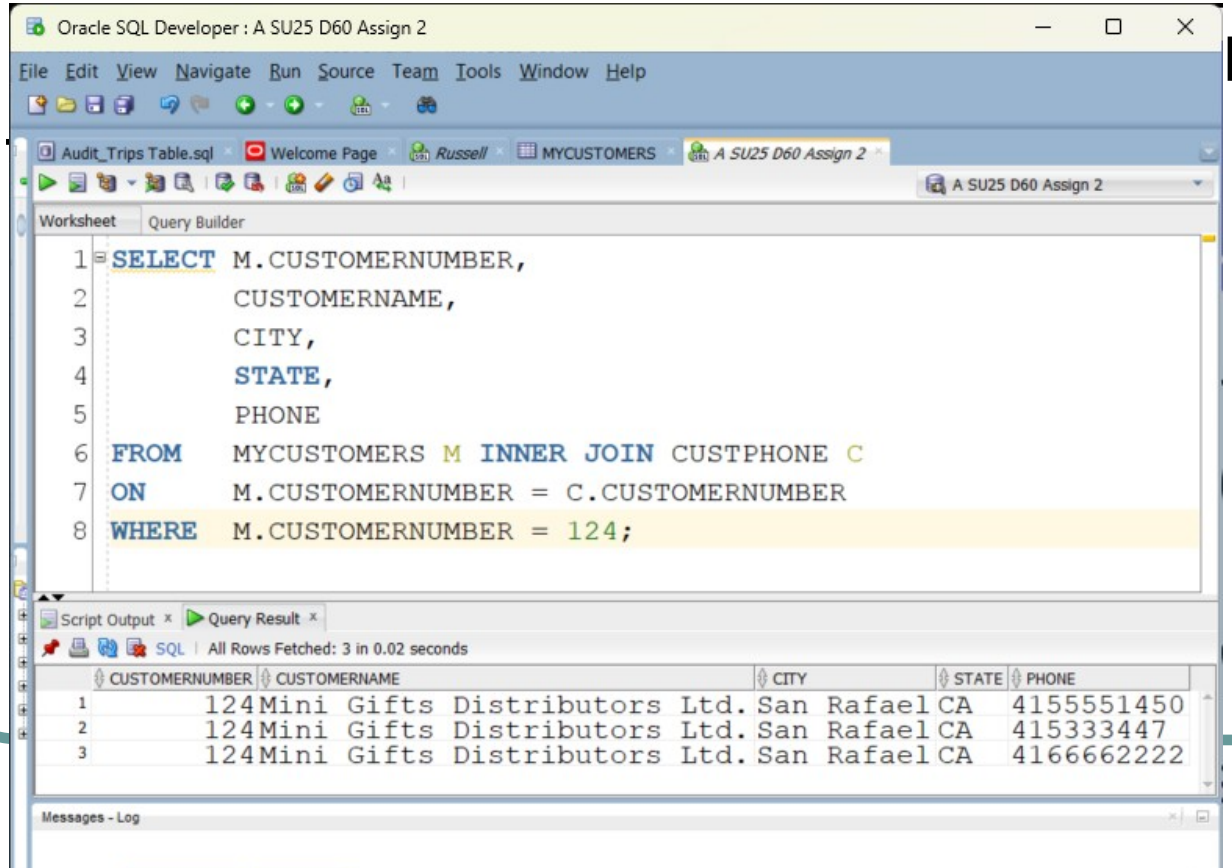


- Remove all phone number columns from the table



RDBMS JOIN

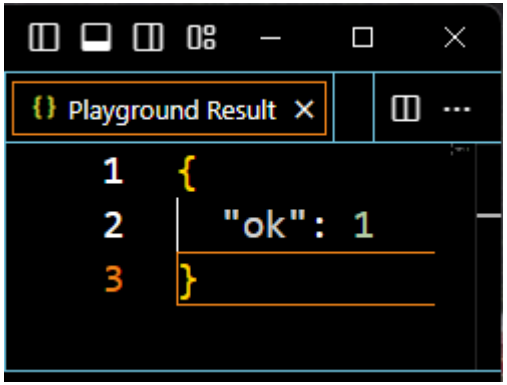
- Get the customer name, city, and state from MYCUSTOMERS
- numbers



NoSQL solution with MongoDB

- MongoDB handles this differently
- That will be shown soon in the MongoDB MyCustomers example

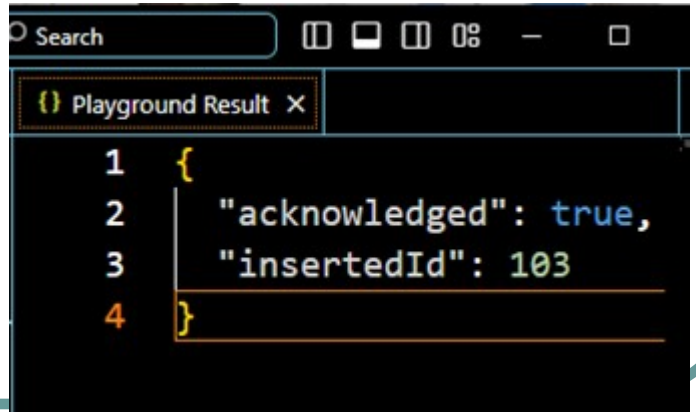
```
const database = 'test';
const collection = 'MyCustomers';

// The current database to use.
use(database);

// Create a new collection.
db.createCollection(collection);
```



MongoDB Demonstration

- insertOne
- The function insertOne adds one document to a collection.
- db.blog.insertOne({"title" : "My Blog Post",
- ... "content" : "Here's my blog post.",
- ... "date" : new Date()})

```
db.MyCustomers.insertOne(
  {       "_id": 103,
          name : "Atelier graphique",
          city : "Nantes",
          country : "France",
          phone : "40.32.2555",
          email : "Atelier_graphique.com",
          creditlimit : 21000
   }
   ) ;
```

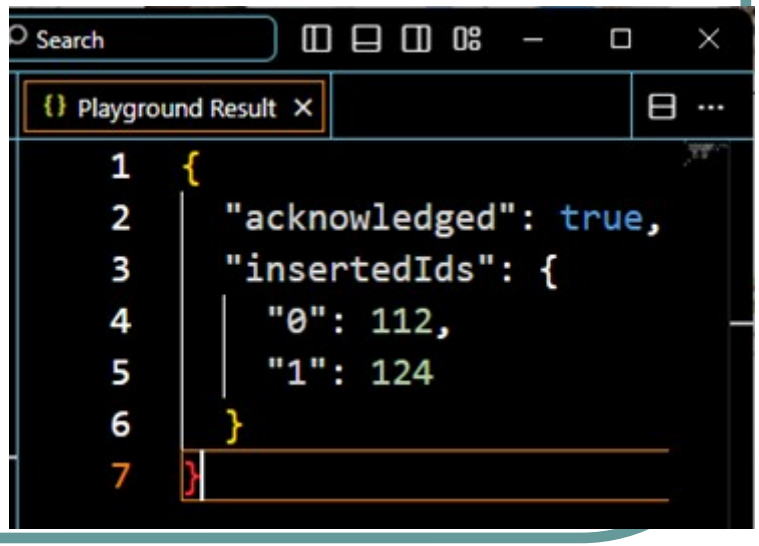


MongoDB Demonstration

- insertMany
- To insert multiple documents, the insertMany function can be used. The function receives an array of documents.
- > db.blog.insertMany([
- … {"title" : "My Blog Post", "content" : "Here's my blog post."},
- … {"title" : "New Post", "content" : "Here's the new blog post."},
- … {"title" : "Public Post", "content" : "Here's the public post."},
-])

- Now lets see how phone is handled with the MyCustomers collection

```
db.MyCustomers.insertMany([
{       "_id": 112,
           name : "Signal Gift Stores",
           city : "Las Vegas",
           country : "USA",
           phone : "7025551838",
           email : "Signal.com",
           creditlimit : 71800
   },
{       "_id": 124,
           name : "Mini Gifts Distributors Ltd.",
           city : "San Rafael",
           country : "USA",
           phone  : "4155551450",
           phone2 : "4153334444",
           phone3 : "4166662222",
           email : "MiniGifts.com",
           creditlimit : 210500
   }
] ) ;
```



MongoDB Demonstration

- Insert Validation
- In an insert operation MongoDB
- Checks the document's basic structure
- The size (must be less than 16 MB)
- Adds an "_id" field if one does not exists
- Invalid data can be easily inserted.
- With an insert operation, if the given collection does not exist, It will be created

MongoDB Demonstration

- Create a Database
- MongoDB creates a database (if it does not exist) when you insert the first document into your database.
- > Use myNewDb
- > db.newCollection.insertOne(“field” : “value”)
- Both myNewDb and newCollection will be created.

- How can you create a collection?
- db.newCollection.insertOne({"_id" : 0});
- The collection is created when you insert your first document in the database
- How do you remove a collection and all of the data inside the collection
- db.newCollection.drop()

MongoDB Demonstration

- Older method for BULK INSERT
- The batch insert can be used to insert multiple documents into a collection by passing an array of documents.
- db.mycustomers.insert([{"_id" : 0}, {"_id" : 1}, {"_id" : 2}])
- { "acknowledged": true,
- "insertedIds": {
- "0": 0,
- "1": 1,
- "2": 2 }}
- DeprecationWarning:
- Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
- We will use db.collectionName.insertMany([

MongoDB Demonstration

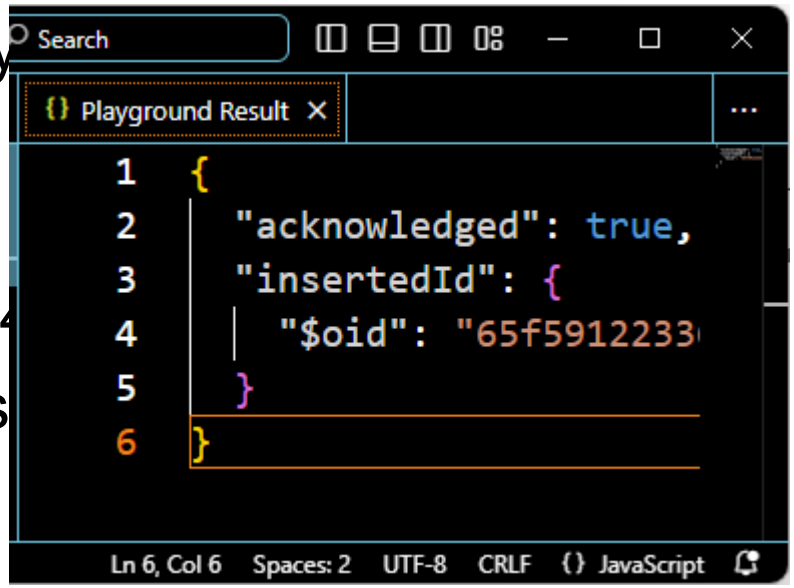
- BULK INSERT
- Example 2:
- > db.foo.insert([{"_id" : 0}, {"_id" : 1}, {"_id" : 1}, {"_id" : 2}])
- In this example, the first two documents will be inserted.
- The insertion of the last two fails because you cannot insert two documents with the same “_id”.
- To ignore errors and insert the rest of the batch the continueOnError option can be used to insert documents after the failure.

MongoDB Demonstration

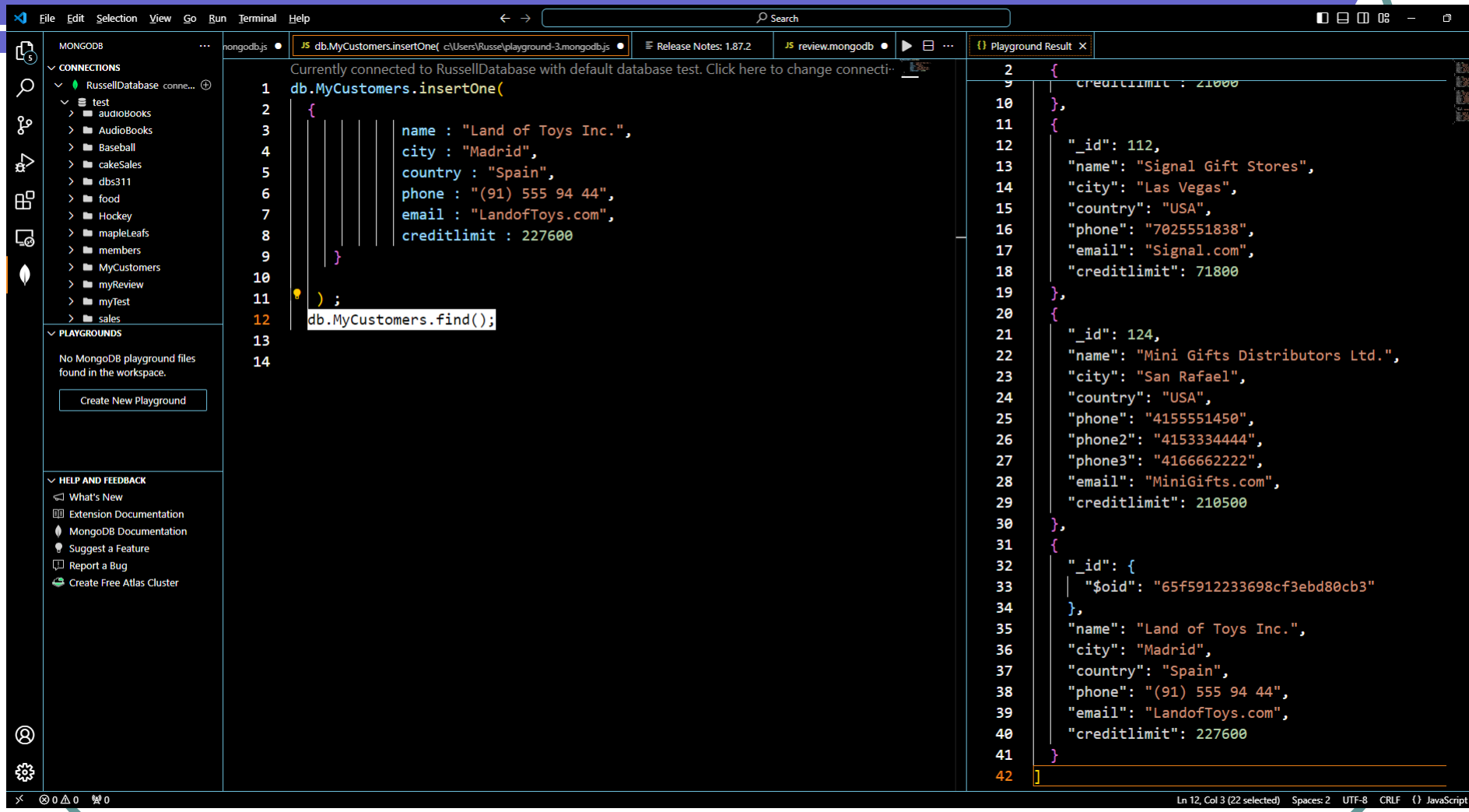
- Remove
- The remove function deletes documents.
- Example 1: > db.blog.remove({})
- The above command removes all documents from the blog collection.
- Example 2: > db.mailing.list.remove({"opt-out" : true})
- This command removes all documents from mailing.list collection if their value of “opt-out” is true.
- The deleted data cannot be recovered after it is removed.
- We will use db.collectionName.deleteMany({});
- Remove is being phased out

When ID is not included

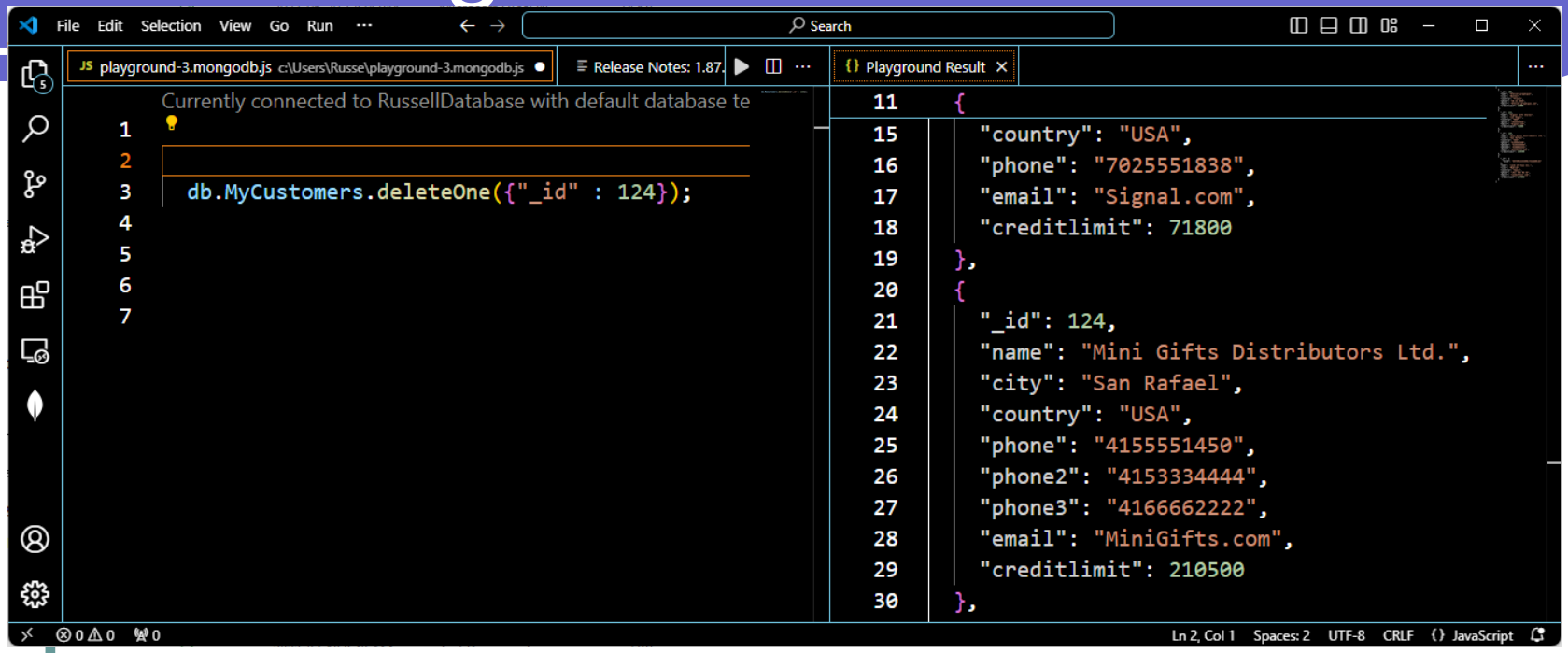
```
db.MyCustomers.insertOne(
  {
        name : "Land of Toy
        city : "Madrid",
        country : "Spain",
        phone : "(91) 555 94
        email : "LandofToys
        creditlimit : 227600
    }
  ) ;
```

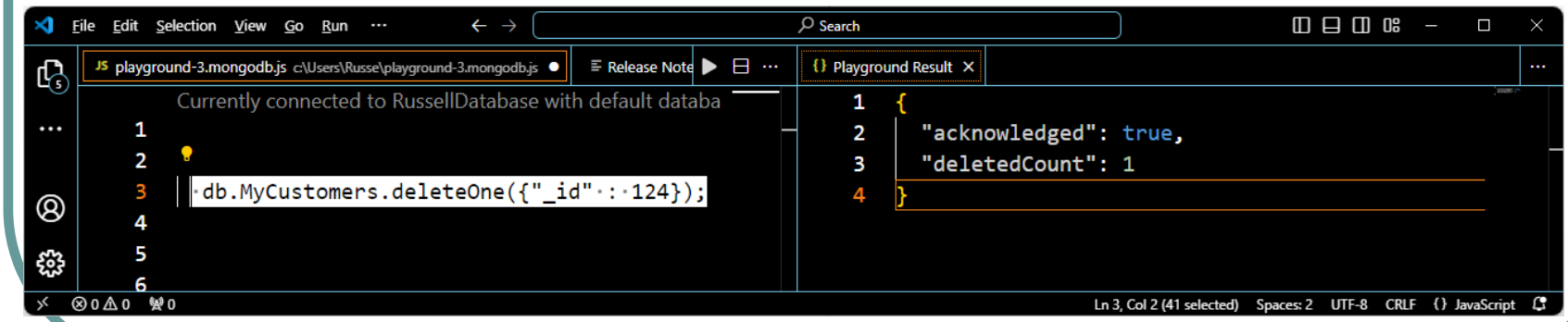


System supplied _id



Removing a document





Searching for Documents

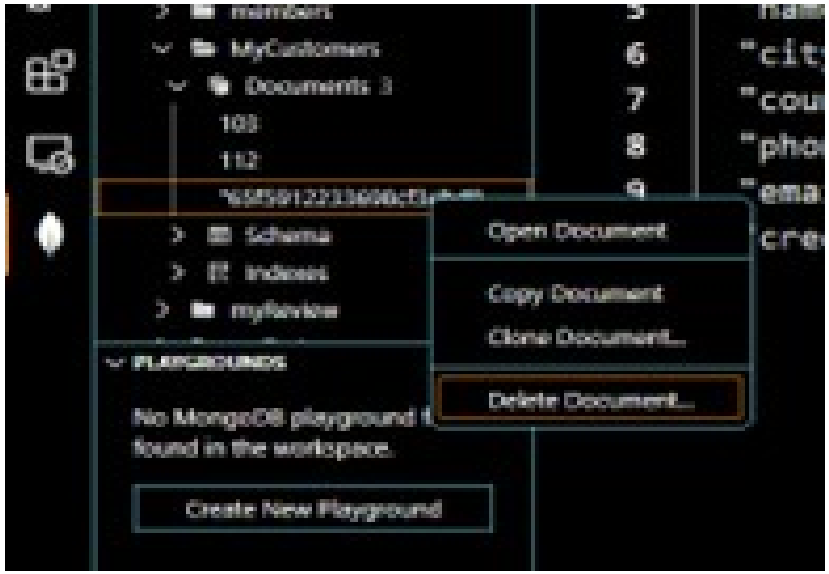
- db.MyCustomers.findOne({"_id" : 124});
- Null
- db.MyCustomers.findOne({"_id" : 112});

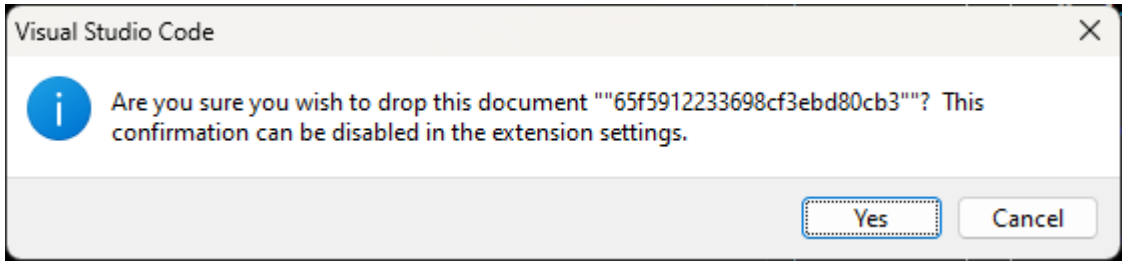
- {
- "_id": 112,
- "name": "Signal Gift Stores",
- "city": "Las Vegas",
- "country": "USA",
- "phone": "7025551838",
- "email": "Signal.com",
- "creditlimit": 71800
- }

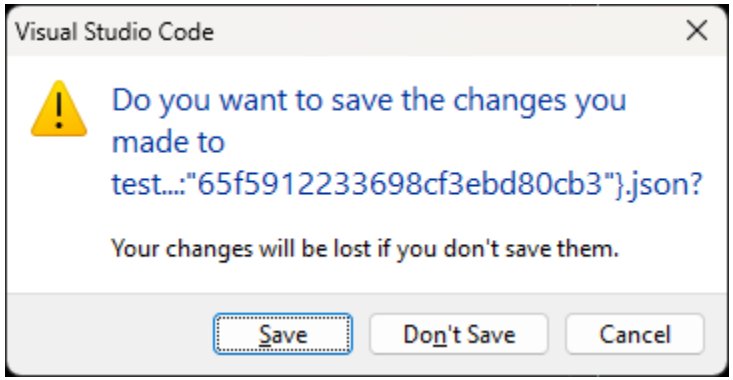
_id not included

- db.MyCustomers.insertOne(
- {
- name : "XXX",
- city : "Nantes",
- country : "France",
- phone : "40.32.2555",
- email : "Atelier_graphique.com",
- creditlimit : 21000
- }
-) ;
- Can be deleted without referring to object ID
- db.MyCustomers.deleteOne({"name" : "XXX"});
- Fine if you want to only delete one document and only one has a name of XXX

Removing a Document using GUI





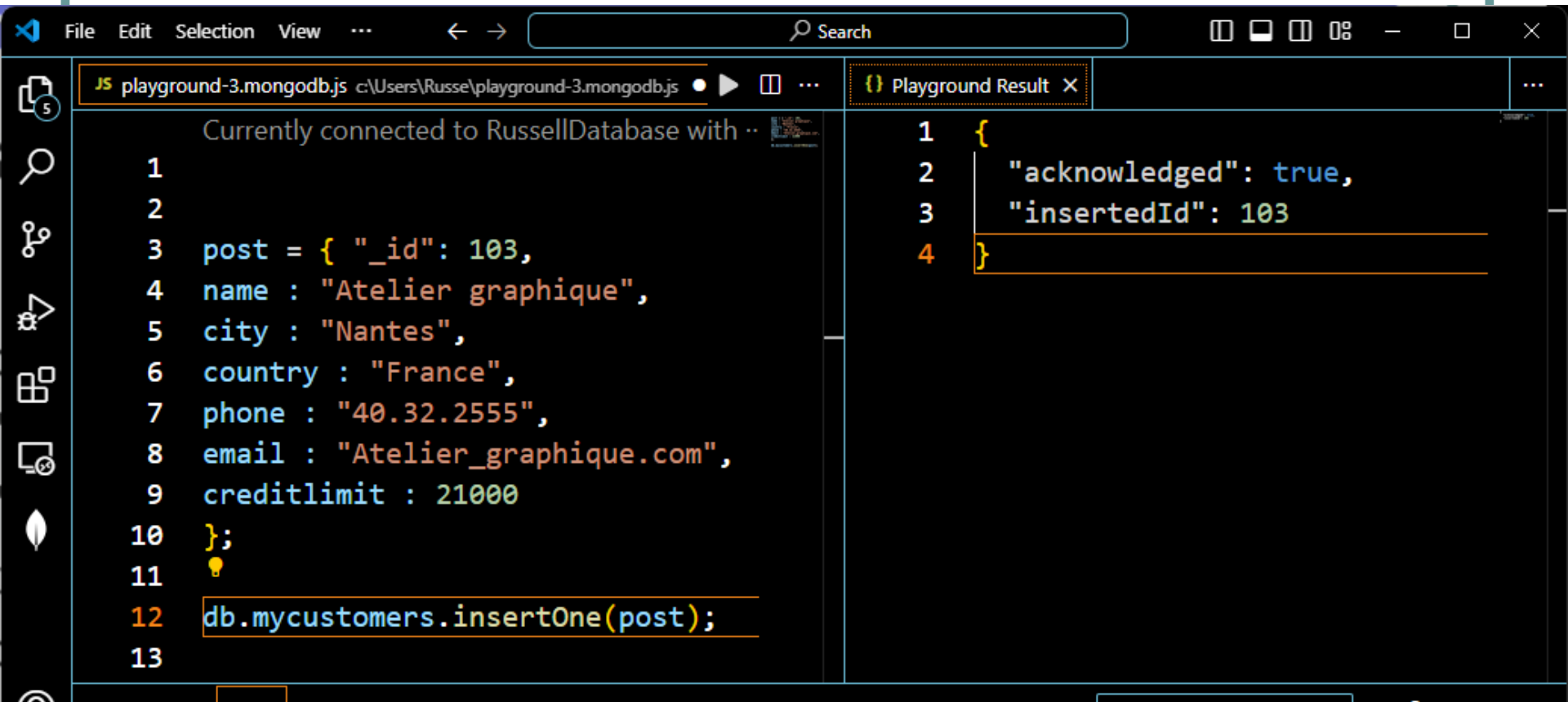


- db.MyCustomers.find({});
- document with generated id is gone
- Other method is to delete all the documents or drop and recreate collection

- Always include a unique _id field with your documents
- As the _id field is immutable, **you cannot change it** and that's by design. This field is all about being the immovable reference to the document

MongoDB Demonstration

- Variables
- A document can be stored to a variable:

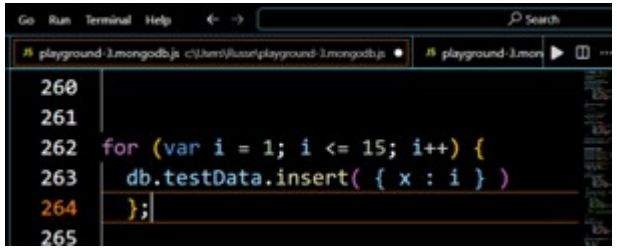


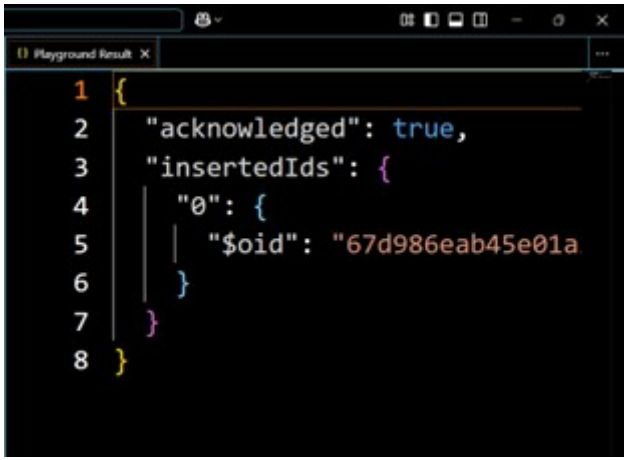
Recap

- What command allows you to put multiple documents into a collection
- insertMany
- What are the opposite commands to insertOne and insertMany?
- deleteOne and deleteMany
- How do you locate all the documents in a collection?
- db.collectionName.find({});
- If a key (in_stock) has a value of 0, then that item is out of stock. How can you locate all the documents with out of stock items in a collection named cars?
- db.cars.find({"in_stock" : 0})
- What if you want to only do something to fhe first of the documents that match
- findOne / deleteOne

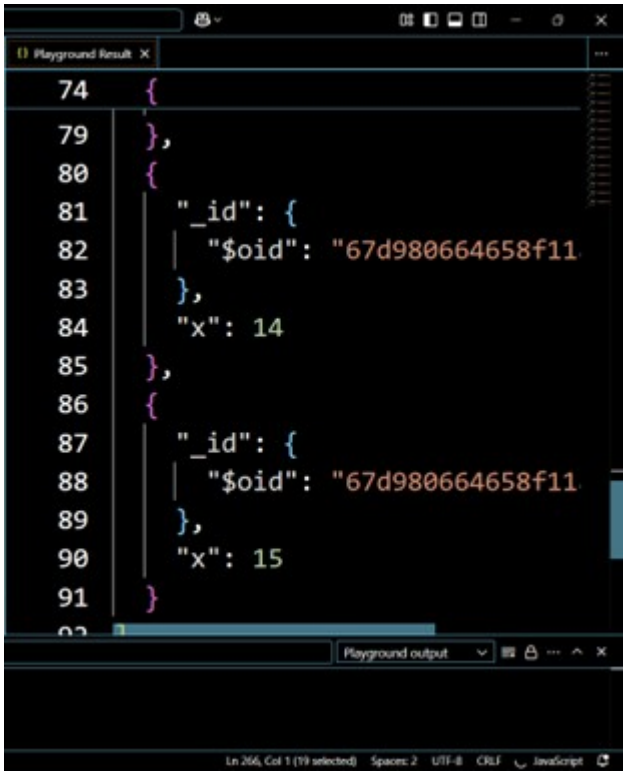
For Loop

- for (var i = 1; i <= 15; i++) {
- db.testData.insert({ x : i })
- };





- db.testData.find();



Drop Collections

- If a collection is dropped, all documents in the collection will be removed.
- `db.testData.drop();`
- This command deletes the db.testData collection.

MongoDB Shell

- MongoDB shell is a full-featured JavaScript interpreter.
  • You can run JavaScript programs in the shell.
- `> x = 200`
- `200`
- `> x / 5;`
- `40`
- We can use the standard JavaScript libraries:
- `> "Hello, World!".replace("World", "MongoDB");`
  `Hello, MongoDB!`
- We can define JavaScript functions:

JavaScript function

```
> function factorial (n) {
... if (n <= 1) return 1;
... return n * factorial(n
- 1);
... }
> factorial(5);
120
lo, MongoDB!
```

MongoDB Demonstration

- Querying a Document
- ad-hoc queries using find or findOne
- Query for
- Range selection
- Set inclusion
- Inequalities
- Cursors

InsertMany

```
db.AudioBooks.insertMany([
{       "_id": 1,
        title : "The Bully Pulpit",
        author : "Doris Kearns Goodwin",
        duration : 2202,
        publisher : "Simon and Schuster Audio",
        narrator : "Edward Hermann",
        datepublished : new ISODate("2013-11-05T14:10:30Z"),
        genre : "Biography"
 },
{       "_id": 2,
        title : "To Sleep in a Sea of Stars",
        author : "Christopher Paolini",
        duration : 2100,
        publisher : "Macmillan Audio",
        narrator : "Jennifer Hale",
        datepublished : new ISODate("2020-09-15T14:10:30Z"),
       genre: "Science Fiction"
},
{
    "_id": 3,
    "title": "Phantoms In The Brain",
    "author": "Sandra Blakeslee",
    "duration": 647,
    "publisher": "Tantor Audio",
    "narrator": "Neil Shah",
    "datepublished":new ISODate("2013-12-24T14:10:30Z")
}
] ) ;
```

MongoDB Demonstration

- Find
- The find function is called to query and see documents in a collection. It returns a subset of documents.
- To see documents in blog collection:
- db.blog.find() Or
- Db.blog.find({})
- The following is the result of calling find function since we have only one document in the blog collection:
- { "_id" : ObjectId("5037ee4a1084eb3ffeef7228"),
- "title" : "My Blog Post",
- "content" : "Here's my blog post.",
- "date" : ISODate("2012-08-24T21:12:09.982Z") }

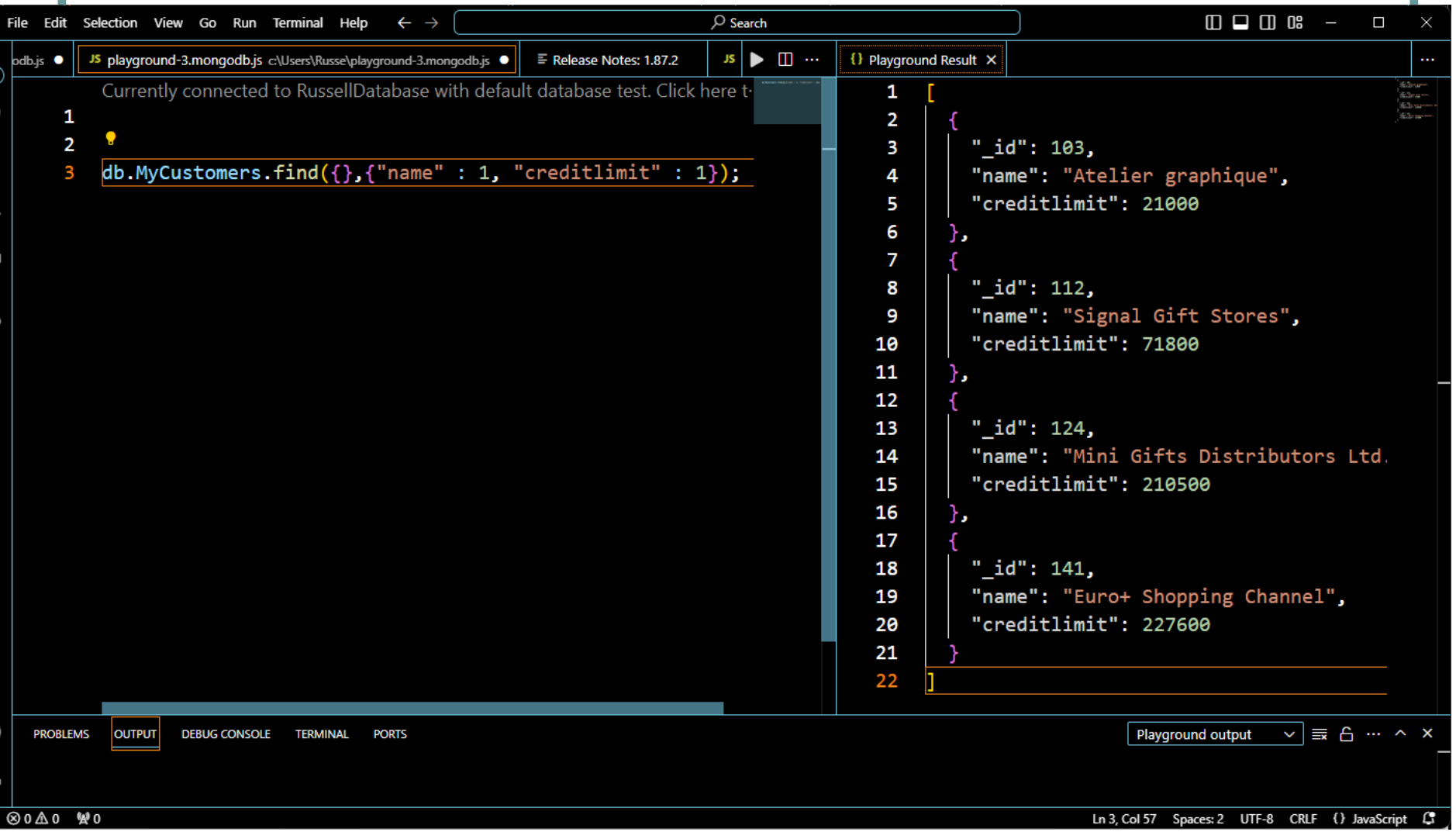
MongoDB Demonstration

- FindOne
- The findOne is used to read and see one document from a collection.
- Example:
- db.blog.findOne()
- {
- "_id" : ObjectId("5037ee4a1084eb3ffeef7228"),
- "title" : "My Blog Post",
- "content" : "Here's my blog post.",
- "date" : ISODate("2012-08-24T21:12:09.982Z")
- }

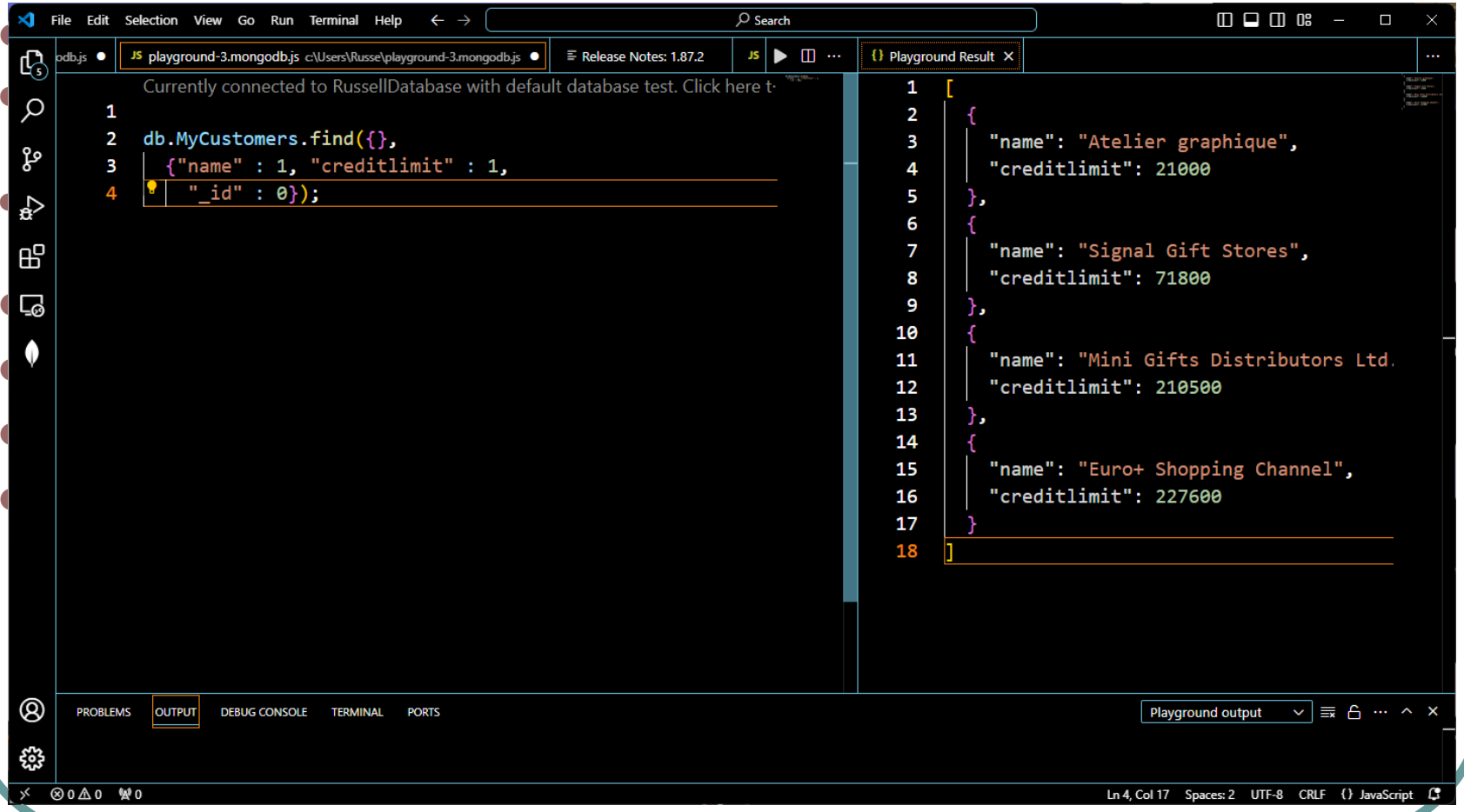
MongoDB Demonstration

- Query with Key/Value Pairs
- We can add key/value pairs to restrict the search.
- Numbers match numbers
- Booleans match Booleans
- Strings match strings
- Find all USA documents :
- db.MyCustomers.find({"country" : "USA"});
- To find documents with multiple conditons, more key/value pairs can be added. We want to find USA customers from Las Vegas:
- db.MyCustomers.find({"country" : "USA", "city" : "Las Vegas"});
- To find items which are out of stock: > db.stock.find({"in_stock" : 0})

MongoDB Demonstration



MongoDB Demonstration



A gotcha

```
db.MyCustomers.insertOne( [
 {      _id : 333,
        name : "ZZZ",
        city : "Nantes",
        country : "France",
        phone : "40.32.2555",
        email : "Atelier_graphique.com",
        creditlimit : 21000
 } ]
 ) ;
```

A gotcha

- db.MyCustomers.find({"name" : "ZZZ"});
- The find looking for a matching document does not work
- Db.MyCustomers.find(); returns the inserted document because it returns all documents

- stored as
- "0": {
- "_id": 333,
- "name": "ZZZ",
- "city": "Nantes",
- "country": "France",
- "phone": "40.32.2555",
- "email": "Atelier_graphique.com",
- "creditlimit": 21000
- },
- "_id": {
- "$oid": "65f70226ea0c1cfe38241279"
- }
- }
-]

A gotcha

- *db.MyCustomers.insertOne([*
- *{ _id : 333,*
- *name : "ZZZ",*
- *city : "Nantes",*
- *country : "France",*
- *phone : "40.32.2555",*
- *email : "Atelier_graphique.com",*
- *creditlimit : 21000*
- *}]*
-) ;
- What was the problem causing this gotcha
- [ ] was the problem – found in insertMany, not insertOne

MongoDB Demonstration

- Query Criteria
- Queries can contain more complex criteria such as
- ranges,
- OR clauses,
- and negation

MongoDB Demonstration

- Comparison Operators

Operators	Description
“$lt”	<
“$lte”	<=
“$gt”	>
“$gte”	>=
“$ne”	<>

- The $ne operator can be used with any type

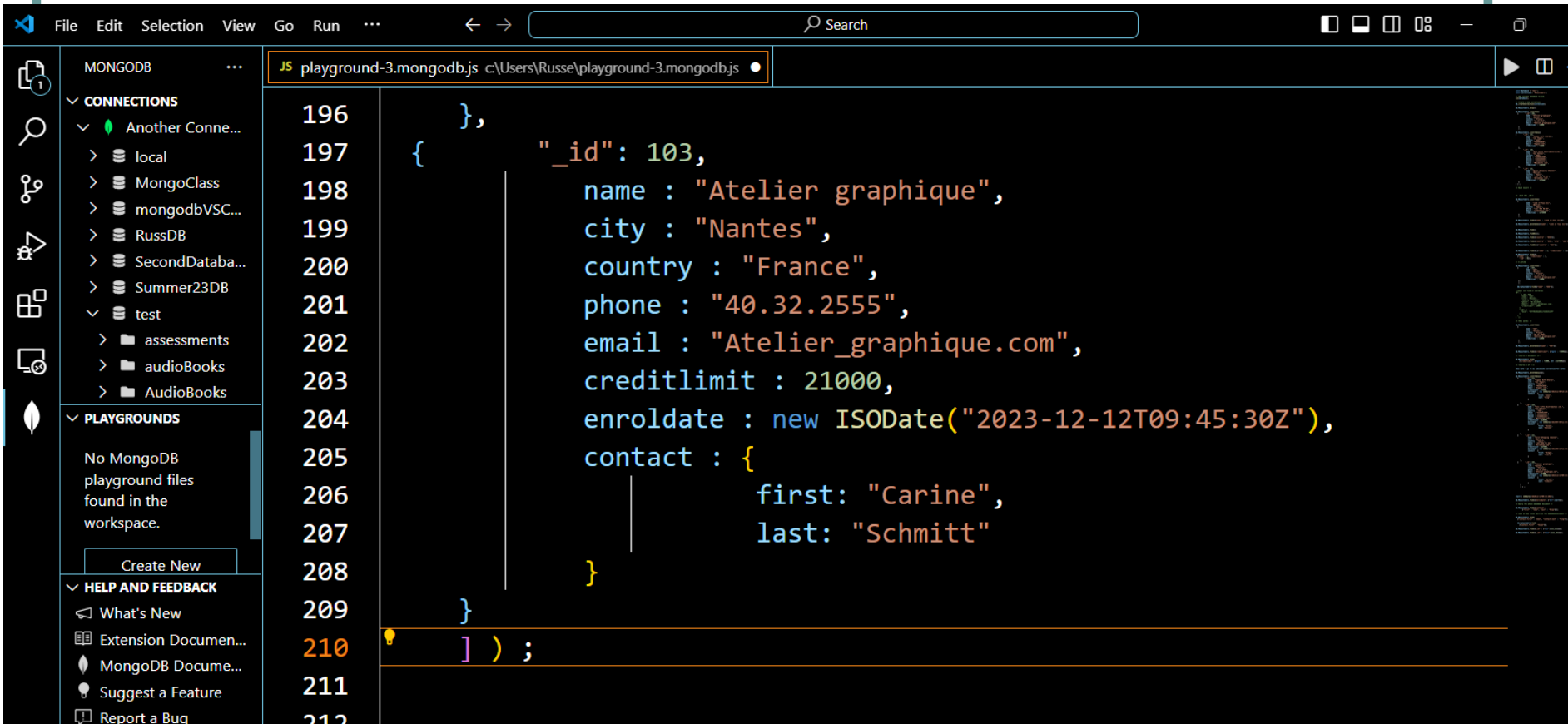
MongoDB Demonstration

- Query Conditionals
- The comparison operators can be combined to create range queries.
- The following query look for users who are between the ages 18 and 30.
- db.users.find({"age" : {"$gte" : 18, "$lte" : 30}})
- Find people who registered before January 1, 2020:
- start = new Date("01/01/2020")
- db.users.find({"registered" : {"$lt" : start}})
- Find all users whose username is not “joe”:
- db.users.find({"username" : {"$ne" : "joe"}}

- db.MyCustomers.deleteMany({});
- Recreate with a date key value pair and an embedded document.

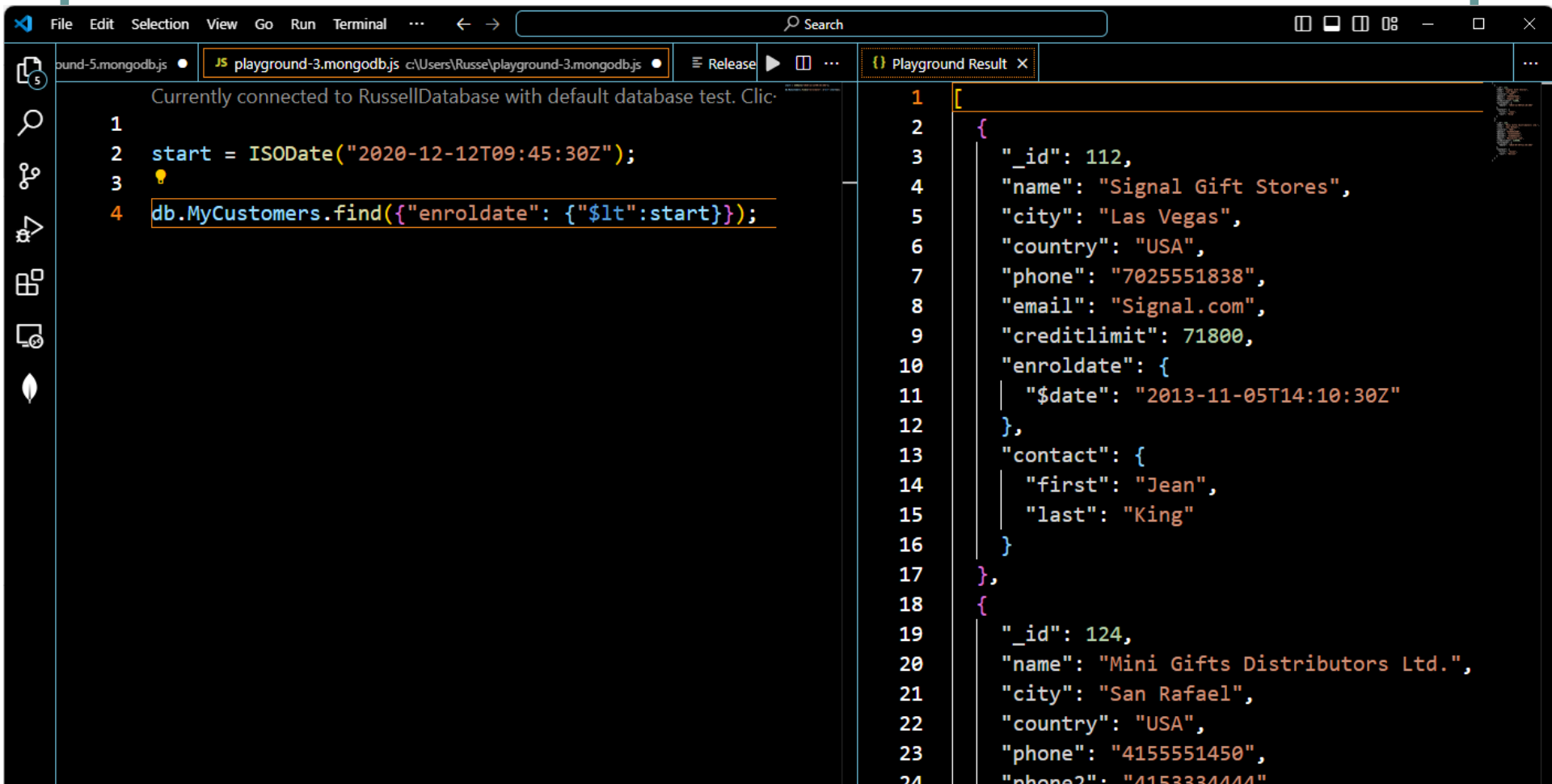
Date and embedded Document

- Date is ISO and embedded document for contacts providing a first and last name



Searching by Date

- Searching for dates less than



MongoDB

- Querying Embedded Documents
- There are two ways of querying for an embedded document:
- querying for the whole document or
- querying for its individual key/value pairs.

MongoDB Embedded Document

- Query the Whole Document
- To query the whole document, consider the following document:
- {
- "name" : {
- "first" : "Joe",
- "last" : "Schmoe"
- },
- "age" : 45
- }
- We can search for someone named Joe Schmoe:
- db.people.find({"name" : {"first" : "Joe", "last" : "Schmoe"}})
- To query an embedded documents for a specific key or keys:
- db.people.find({"name.first" : "Joe", "name.last" : "Schmoe"})

MongoDB Demonstration

- Query the whole embedded document
- db.MyCustomers.find({"contact" :
- {"first" : "Jean", "last" : "King"}});
- Look at key value pairs in the embedded document
- db.MyCustomers.find(
- {"contact.first" : "Jean", "contact.last" : "King"});
- db.MyCustomers.find(
- {"contact.first" : "Susan"});

MongoDB

- Query Specific Keys
- Consider the following document: { "comments" : [
- {
- "author" : "joe",
- "score" : 3,
- "comment" : "nice post"
- },
- { "author" : "mary",
- "score" : 6,
- "comment" : "terrible post" }] }
- Query
- db.blog.find({"comments" : {"author" : "joe", "score" : {"$gte" : 5}}})
- The above query is not correct. Embedded document matches have to match the whole document.

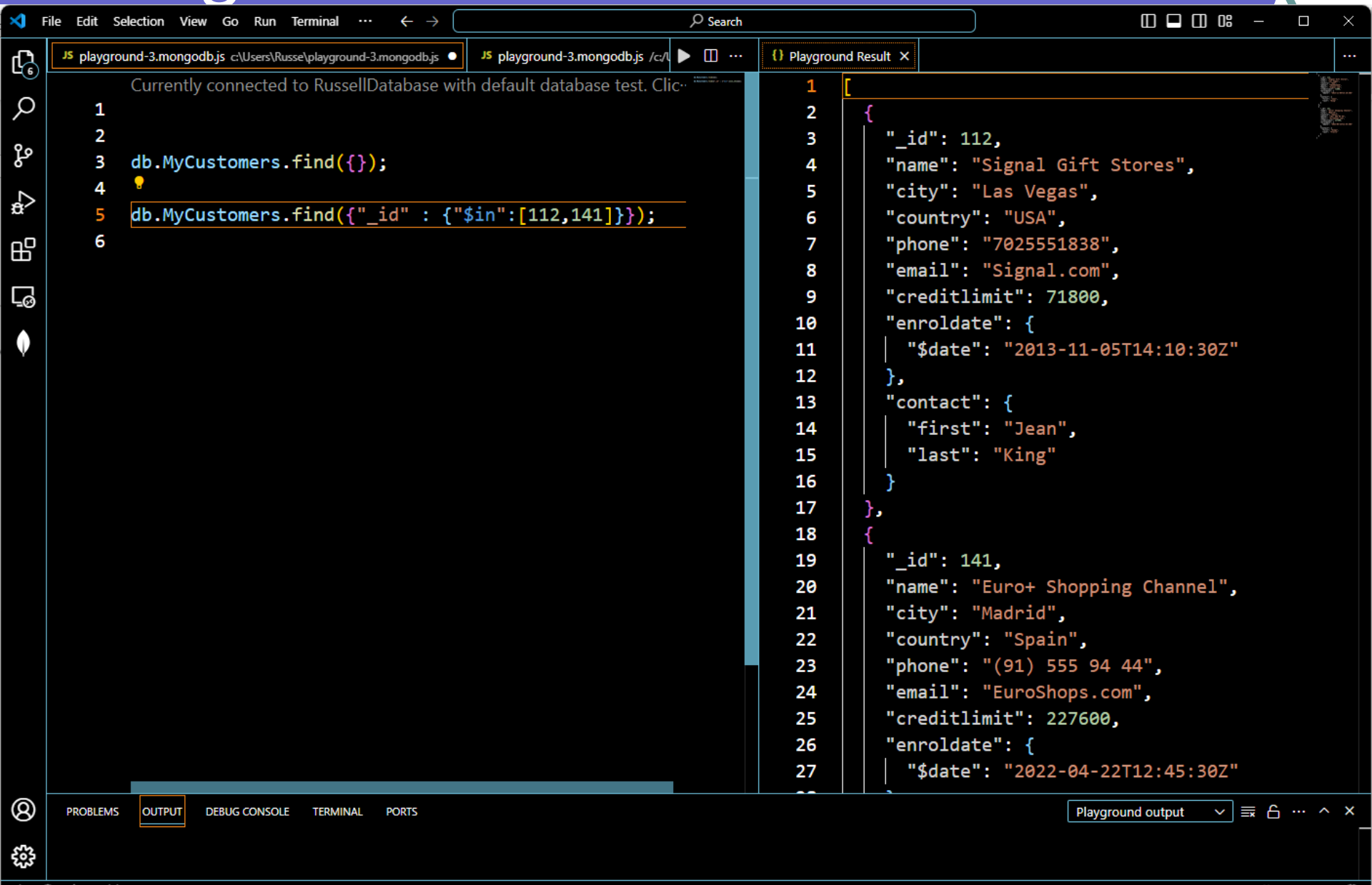
MongoDB

- OR Queries
- There are two ways to do an OR query in MongoDB.
- To query documents based on a single key.
- “$in”
- To check different criteria or different keys. (To combine conditions)
- “$or”

MongoDB

- “$in” Operators
- The “$in” operators is used to search a variety of values for a single key.
- To match more than one value for a single key, use an array of values with the “$in” operator.
- db.raffle.find({"ticket_no" : {"$in" : [725, 542, 390]}})
- The value of the “ticket_no” key is compared to three values.
- "$in" is very flexible and allows you to specify criteria of different types as well as values.
- db.users.find({"user_id" : {"$in" : [12345, "joe"]})
- The following statements are equivalent:
- {ticket_no : {$in : [725]}}
- {ticket_no : 725}

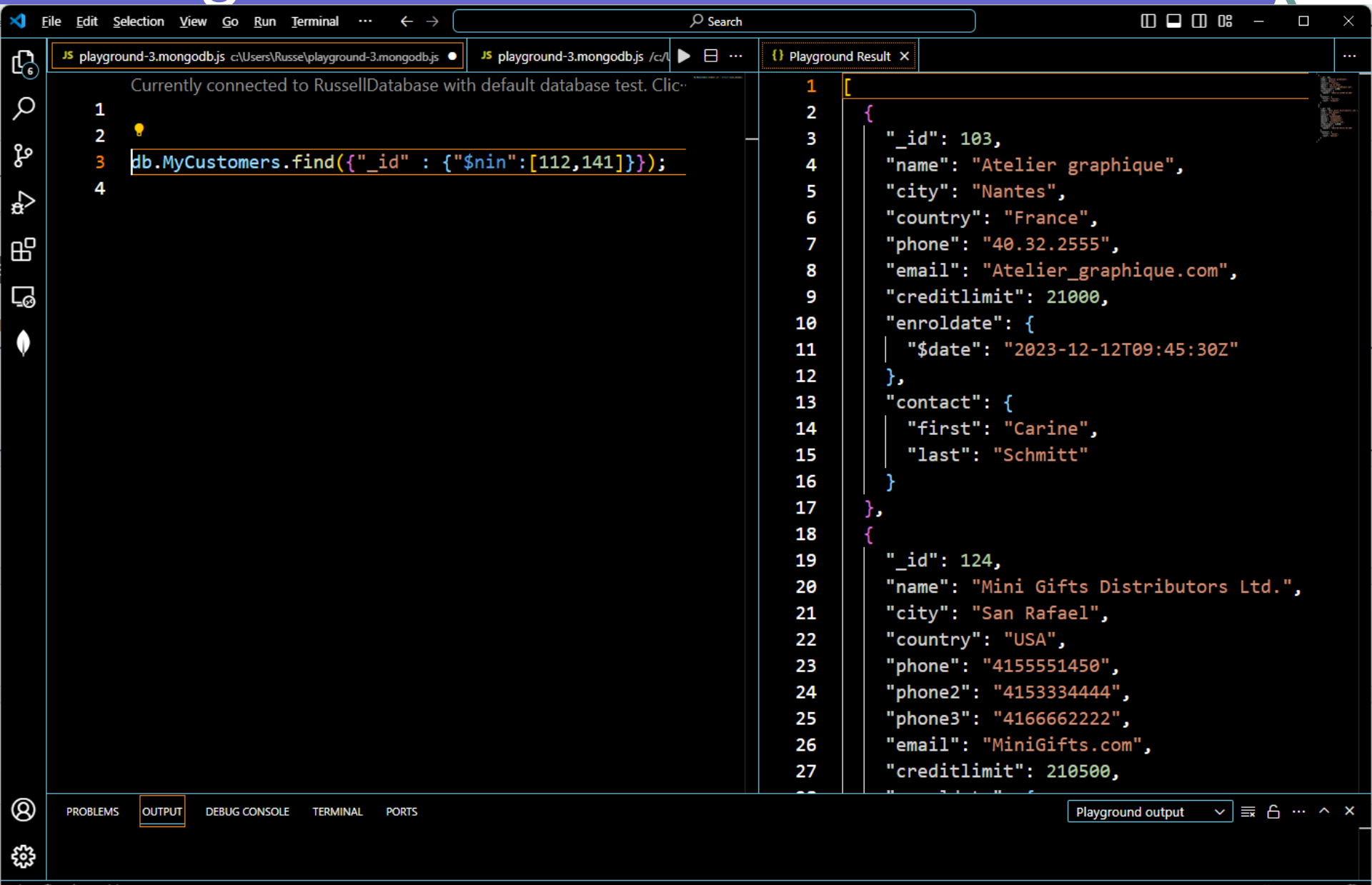
MongoDB Demonstration



MongoDB

- “$nin” Operator
- The “$nin” operator returns documents that don’t match any of the criteria in the array.
- db.raffle.find({"ticket_no" : {"$nin" : [725, 542, 390]}})
- The above query returns people who do not have any tickets with the given numbers.

MongoDB Demonstration



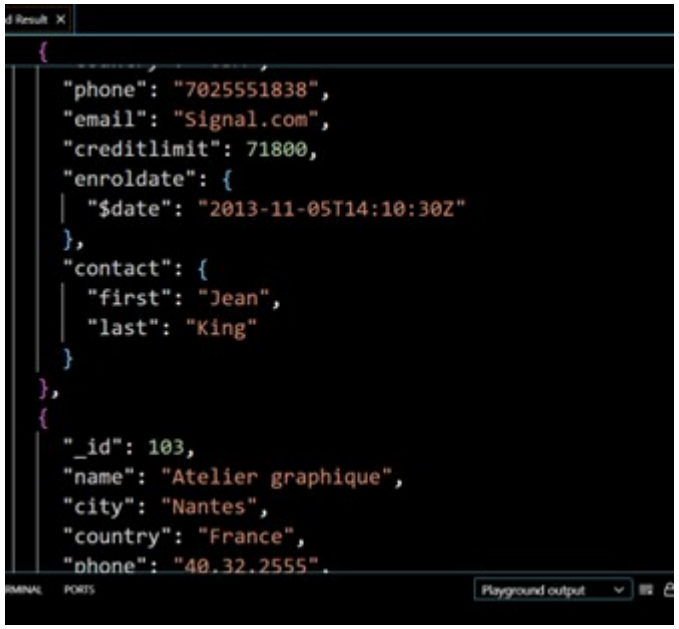
MongoDB Demonstration

- “$or”/”$and” Operator
- The “$or” operator is used to check an array of possible criteria. The query returns the document if either condition is true.
- db.raffle.find({"$or" : [{"ticket_no" : 725}, {"winner" : true}]})
-
- db.raffle.find({"$or" : [{"ticket_no" : {"$in" : [725, 542, 390]}},
- {"winner" : true}]})

- See the following example of “$and”:
- db.users.find({"$and" : [{"x" : {"$gt" : 1}}, {"x" : {“$lt” : 4}}]})

$or

- db.MyCustomers.find({"$or":[{"contact.first":"Jean"},
- {"city" : "Nantes"}]});



MongoDB Demonstration

- “$not” Operator
- The “$not” operator can be applied on top of any condition.
- db.inventory.find({ price: { $not: { $gt: 1.99 } } })
- This query will select all documents in the inventory collection where:
- the price field value is less than or equal to 1.99 or
- the price field does not exist
- { $not: { $gt: 1.99 } } is different from the $lte operator.
- { $lte: 1.99 } returns only the documents where price field exists and its value is less than or equal to 1.99.

- Lab Collection
- Based on one of the models handed out
- I have audiobooks
- Minimum 7 key value pairs
- Include numeric, character and date
- Demonstrate how MongoDb differs from a relational DBMS by having one document with 8 key value pairs

- Have all your statements ready to run.
- Demonstrate some searches that return specified key value pairs, only match some of the documents, omit the _id
- Demonstrate both use of or / and

Lab

- Lab 8 has been been posted